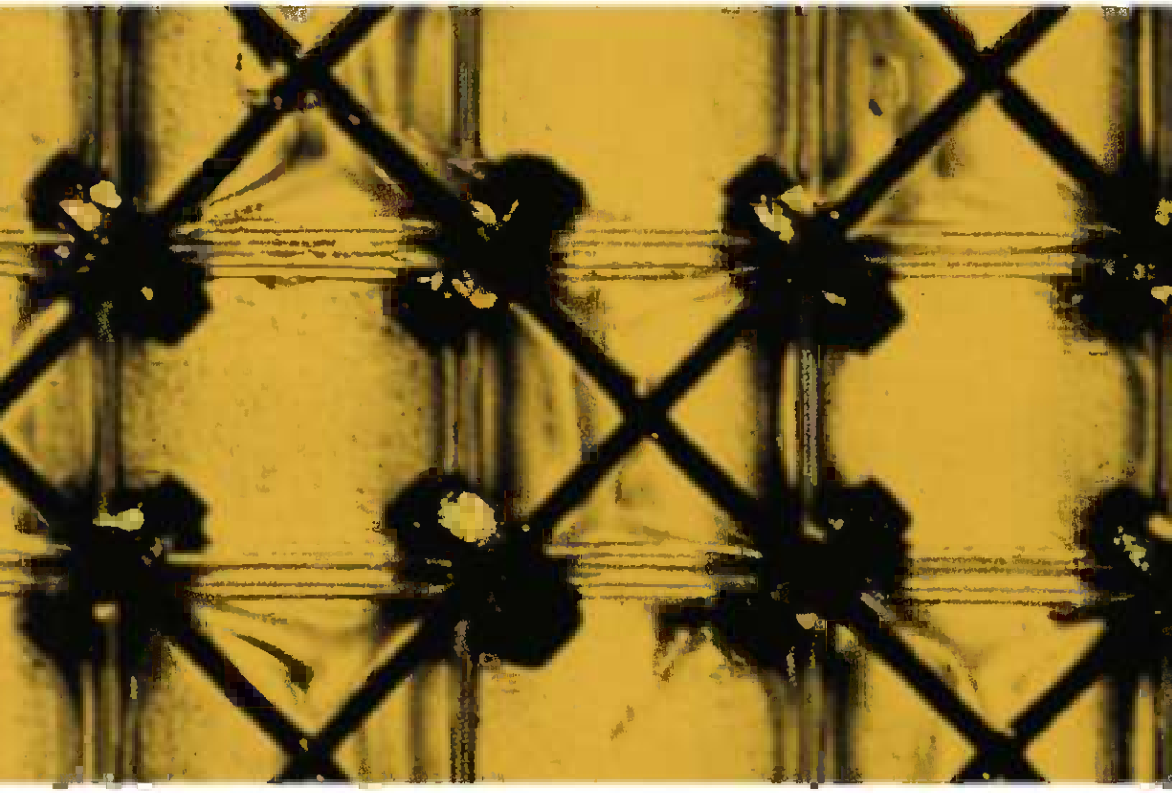
Computer + Programmieren = Elektronische Datenverarbeitung

K.W. Blutke



GABLER FACH BUCH

Mit einem Geleitwort von Prof. Dr.-Ing. K. Steinbuch

Blutke · Computer + Programmieren = Elektronische Datenverarbeitung

GABLER
FACH
BUCH

Klaus Wilhelm Blutke

Computer + Programmieren = Elektronische Datenverarbeitung

3. Auflage

GABLER

ISBN 978-3-409-96051-9 ISBN 978-3-663-13670-5 (eBook)
DOI 10.1007/978-3-663-13670-5

© 1979 Springer Fachmedien Wiesbaden

Ursprünglich erschienen bei Betriebswirtschaftlicher Verlag Dr. Th. Gabler KG, Wiesbaden 1979.
Umschlaggestaltung: Horst Koblitz, Wiesbaden
Satz: H. Erhart Henniger, Satzbetriebe, Wiesbaden

Alle Rechte vorbehalten. Auch die fotomechanische Vervielfältigung des Werkes (Fotokopie, Mikrokopie) oder von Teilen daraus bedarf der vorherigen Zustimmung des Verlages.

Geleitwort

Es gibt wenige Vorgänge, welche unsere Lebensrealitäten so stark verändern wie der Computer. Deshalb ist eine didaktisch geschickte Einführung in das Computerwesen von größter Wichtigkeit. K.W. Blutke hat hier die Probleme der Hardware und Software in geschickter und praxisnaher Weise dargestellt, so daß sie jeder Interessierte – auch ohne spezielle Vorkenntnisse – verstehen kann.

Der Darstellungsstil zeigt einerseits gründliche Sachkenntnis und andererseits Einfühlungsvermögen in das Denken interessierter Außenstehender. Daß technische Erklärungen und Anwendungsbeispiele geschickt miteinander verwoben sind, ist ein besonderer Reiz der Darstellung. Vor allem die zahlreichen und wohlplazierten Abbildungen sind geeignet, die Scheu des Anfängers zu überwinden.

Es ist erfreulich, daß diese vorzügliche Darstellung nunmehr – überarbeitet und auf den neuesten Stand der Technik gebracht – in dritter Auflage erscheint.

Ich wünsche der neuen Auflage die gebührende Beachtung und Verbreitung.

K. Steinbuch

Vorwort und Hinweis zur 3. Auflage

Begriffe wie elektronische Datenverarbeitung (EDV), Computer und Programmieren gehören heute bereits zum allgemeinen Sprachschatz.
Für viele von uns sind aber solche Begriffe bisher leider nur Schlagworte geblieben; doch diesem Zustand soll dieses Buch abhelfen. Dabei werden keine speziellen Vorkenntnisse vorausgesetzt. Der gesamte Stoff wird anhand verständlicher Beispiele vermittelt:
Sie erfahren, wie ein Computer programmtechnisch arbeitet und welche Möglichkeiten er im Rahmen der elektronischen Datenverarbeitung bietet; Sie werden Sinn und Zweck der Programmierung erfassen und in die Lage versetzt, kleinere Programme selbst zu konzipieren.
Mit dieser Einführungsschrift möchte ich jene Interessenten ansprechen, die sich fundierte EDV-Grundkenntnisse aneignen möchten, die sich aus beruflichen Gründen der EDV zuwenden wollen und die in ihrer Schulzeit mit EDV in Berührung kommen.
Unter den Programmiersprachen (ASSEMBLER, COBOL, FORTRAN, ALGOL, PL/1 etc.) halte ich jene für besonders geeignet, das Verständnis für die Programmierung und die Arbeitsweise des Computers zu fördern, die dem sogenannten „Zwei-Adress-Prinzip" unterliegen. Aus diesem Grunde wählte ich zur Erläuterung die Programmiersprache ASSEMBLER weitverbreiteter IBM-, SIEMENS- und UNIVAC-Systeme u. a. m.
Unabhängig davon werden Sie aber auch befähigt, Spezialliteratur anderer Datenverarbeitungsanlagen lesen und verstehen zu können.
In der vorliegenden dritten Auflage hat dieses Buches insbesondere durch die neuen Kapitel „Speicherungs- und Verarbeitungsformen von Magnetplatten-Dateien" und „Die Programmbibliothek auf der Magnetplatte" eine wesentliche Erweiterung erfahren. Es soll dadurch die Kenntnisse eines EDV-ABC's abrunden helfen.
Der Bildanhang wurde aktualisiert. In diesem Zusammenhang danke ich den Firmen AEG-TELEFUNKEN, AMDAHL, DIGITAL EQUIPMENT, HEWLETT PACKARD, HONEYWELL BULL, IBM, ICL, KIENZLE, NCR, NIXDORF, SIEMENS und SPERRY UNIVAC für die Genehmigung zur Wiedergabe verschiedener Abbildungen und Fotos.
Besonderer Dank gebührt auch dem Verlag für seine Bemühungen.

Inhaltsverzeichnis

1. Was ist ein Programm? 11
2. Die Lochkarte 24
3. Ein Computer liest Lochkarten 28
4. Der Hauptspeicher 31
5. Der Computer erhält seine erste Arbeit 34
6. Adressierung im Hauptspeicher 50
7. Dateien 54
8. Das symbolische Programm 60
9. Die Umwandlung eines symbolischen in ein echtes Programm 73
10. Zahlensysteme 76
11. Darstellungsmöglichkeiten für Hauptspeicherinformation 80
12. Dezimale und duale Rechenoperationen 83
13. Das Dualzahlenaddierwerk des Computers 100
14. Ein „Gruppenwechsel", porgrammtechnisch betrachtet 119
15. Instruktionsformate 129
16. Tabellen-Lesen 141
17. Sortieren und Mischen 146
18. Verschiedene Ein/Ausgabe-Einheiten 151
19. Der Drucker 169
20. Magnetband-Einheiten 176
21. Magnetplatten-Einheiten 192
22. Speicherungs- und Verarbeitungsformen von Magnetplatten-Dateien 202
23. Die Programmbibliothek auf der Magnetplatte 246
24. Die problemorientierten Programmiersprachen 258
25. Computergenerationen und Betriebssysteme 261

Bildanhang 268

Bildtafeln

Abbildungen von Bauelementen, Baugruppen und anderen Einzelheiten der verschiedenen Computergenerationen

Baugruppen der verschiedenen Computergenerationen . . . . . Tafel 1
Elektrische Bauelemente der ersten drei Computergenerationen . . . Tafel 2
Extrem kleine Magnetringkerne . . . . . Tafel 3
Magnetkernspeicherebene . . . . . Tafel 4
Mikrobaugruppe der SLT-Technik . . . . . Tafel 5
Mehrlagige gedruckte Schaltung . . . . . Tafel 6
Monolith im Größenvergleich zu Einzellern . . . . . Tafel 7
Silizium-Wafer und Chips in LSI-Technik . . . . . Tafel 8
MTL-Grundbauelement . . . . . Tafel 9
Prüfung einer Datenverarbeitungsanlage . . . . . Tafel 10
Halbleiterspeicher . . . . . Tafel 11
Laserdrucker . . . . . Tafel 12
Schreib/Leseköpfe eines Magnetplattenspeichers . . . . . Tafel 13

Abbildungen elektronischer Datenverarbeitungsanlagen

IBM Serie / 1 . . . . . Tafel 14
SIEMENS-EDV-Redaktionssystem COSY 200 . . . . . Tafel 14
IBM System Serie / 1 . . . . . Tafel 15
Programmierbarer Tischrechner HP 9825A (HEWLETT-PACKARD) . Tafel 15
Doppelprozessorsystem UNIVAC 1106 . . . . . Tafel 16
Datensichtstation UNISCOPE 100 . . . . . Tafel 16
Datenendgerät AEG-Telefunken . . . . . Tafel 17
EDV-System KIENZLE 6100 . . . . . Tafel 17
NCR-COSYNA-Computersystem . . . . . Tafel 18
NIXDORF-Datentelefon . . . . . Tafel 18
ICL-Terminalprozessor 7502/15 . . . . . Tafel 19
HONEYWELL BULL Terminalcomputer 61/40 . . . . . Tafel 19

1. Was ist ein Programm?

Ausführungsbeispiel Nr. 1

Maxl, Lagerarbeiter einer Holzhandlung, hat einen Waggon mit Brettern vier verschiedener Holzarten (Eiche, Teak, Palisander, Fichte) zu entladen und die Bretter jeder Holzart auf einem bestimmten Platz zu stapeln (Abb. 1).
Wie geht er vor?

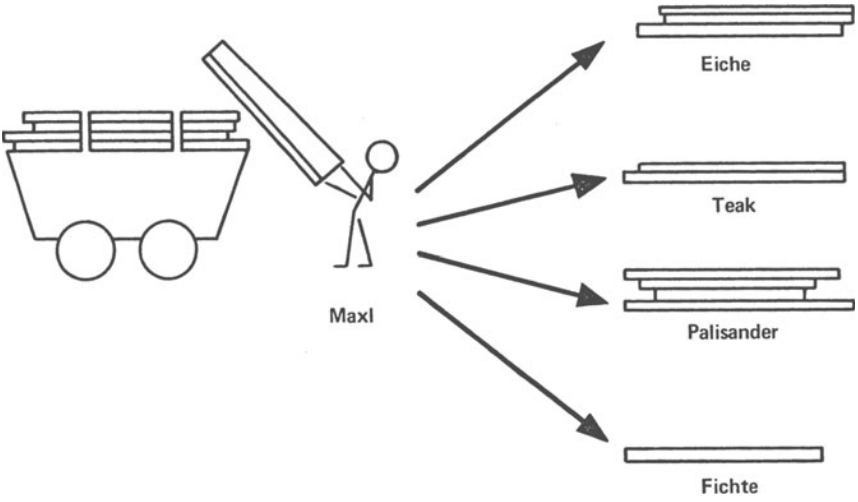
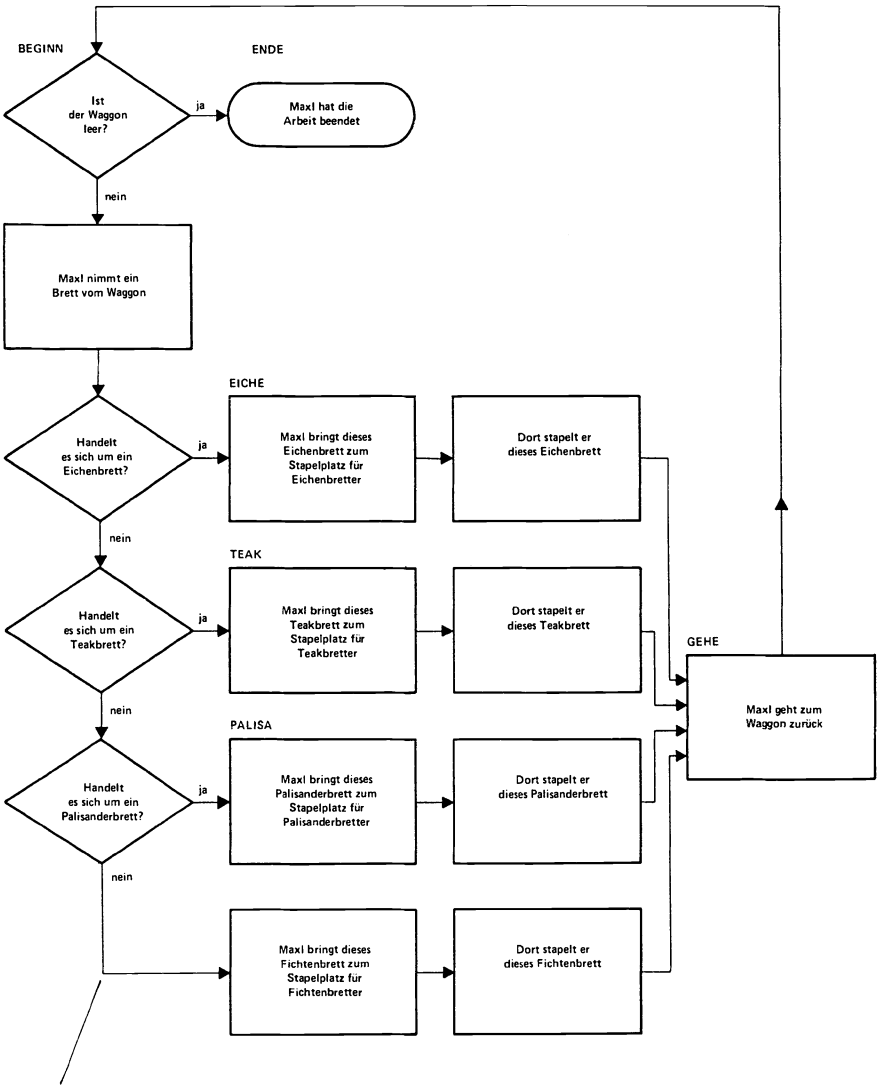


Abb. 1

a) Er nimmt ein Brett vom Waggon,
b) stellt dessen Holzart fest,
c) geht mit diesem Brett zum vorgeschriebenen Stapelplatz,
d) legt dort das Brett ab,
e) geht zurück zum Waggon und beginnt wieder bei a), solange noch Bretter auf dem Waggon sind.

Der Ablauf (das Programm) einer solchen Arbeit läßt sich als Schema in einem Blockdiagramm (Programmablaufplan) wie in Abb. 2 darstellen. Darin sind die Fragen stets so zu formulieren, daß sie nur mit ja oder nein beantwortet werden können.
Erläuterungen zu Abb. 2:
Die einzelnen Schritte des Programmes folgen immer in Pfeilrichtung aufeinander. Wird eine Frage im Blockdiagramm mit ja beantwortet, so läuft das Programm im Ja-Zweig, andernfalls im Nein-Zweig weiter (Abb. 3).



Hier kann es sich nur um ein Fichtenbrett handeln, da nach Eichen-, Teak- und Palisanderbrettern bereits gefragt wurde.

Abb. 2

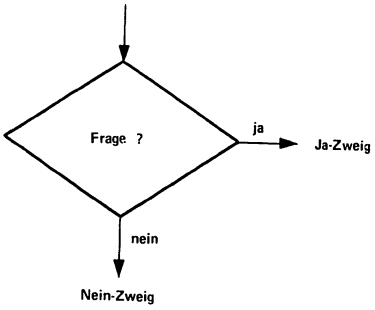


Abb. 3

Das Programm beginnt mit der Frage

Ist der Waggon leer?

Zu dieser Frage kehrt es auch immer wieder zurück. Solange sich Bretter auf dem Waggon befinden, kann Maxl seine Arbeit aufnehmen bzw. fortsetzen. Ergreift er ein Palisanderbrett, so wird das Blockdiagramm gemäß der gestrichelten Linie durchlaufen. Wieder angelangt an der Frage

Ist der Waggon leer?

muß erst einmal entschieden werden, ob noch weitere Bretter abzuladen sind. Ist dies der Fall, so kann Maxl erneut ein Brett ergreifen und die Arbeit fortsetzen. Handelte es sich um ein Fichtenbrett, müßte das Programm dem strich-punktierten Weg folgen. Usw. usw. . . . .
Nach dem Abladen und Stapeln des allerletzten Brettes würde schließlich die Frage

Ist der Waggon leer?

bejahend beantwortet werden. Das wäre aber gleichbedeutend mit dem Arbeitsende.
Die graphische Darstellung eines Arbeitsablaufes in Form eines Blockdiagramms ist sehr übersichtlich. Ausgehend von diesem Blockdiagramm kann aber auch derselbe Arbeitsablauf in einem Zeilen-Schema (Abb. 4) erfaßt werden.
Jeder Block des Blockdiagramms stellt einen Programmschritt, eine sogenannte Instruktion dar. Bestimmte Instruktionen des Blockdiagramms müssen einen Namen erhalten, der frei wählbar ist:

1. Die Instruktion, mit der das Programm beginnt
   (In Abb. 2 wurde dafür der Name BEGINN vergeben)

2. Jede Instruktion, die am Anfang eines Ja-Zweiges steht
   (In Abb. 2 lauten die entsprechenden Namen ENDE, EICHE, TEAK und PALISA)

3. Jede Instruktion, zu der mehr als von einer Stelle des Programmes verzweigt wird, zu der also mehr als ein Pfeilweg führt
(In Abb. 2 wäre dies die Instruktion namens GEHE)

Für das Zeilen-Schema steht ein Formular bereit (Abb. 4), das eine Unterteilung in die Spalten Name und Instruktion vorsieht. Zum Erstellen des Zeilen-Schemas werden unter Beachtung gewisser Richtlinien die Eintragungen des Blockdiagrammes herangezogen, eine Tätigkeit, die man schlechthin auch als Abschreiben bezeichnen kann:
Im Blockdiagramm (Abb. 2) beginnt das Programm mit der Instruktion

BEGINN Ist der Waggon leer?

Die entsprechende Eintragung im Zeilen-Schema (Abb. 4) lautet ebenfalls

BEGINN Ist der Waggon leer?

Fragen ziehen stets die Antwort „ja" oder „nein" nach sich; folglich wird das Programm entweder im Ja- oder Nein-Zweig dieser Frage fortgesetzt.
Für das Zeilen-Schema gilt die Vorschrift, daß nach einer Frage immer die Instruktion zu benennen ist, bei der das Programm für die Antwort „ja" fortzuführen ist.

BEGINN Ist der Waggon leer?
Programm mit der Instruktion ENDE fortführen bei Antwort ja

Im Anschluß daran werden die Instruktionen des Nein-Zweiges dieser Frage aufgeführt:

BEGINN Ist der Waggon leer?
Programm mit der Instruktion ENDE fortführen bei Antwort ja
Maxl nimmt ein Brett vom Waggon
usw.

Wird die Frage nach dem leeren Waggon bejaht, so endet die Fortführung des Programmes bei der Instruktion ENDE (diese Instruktion erscheint an einer anderen Stelle des Zeilen-Schemas). Bei einer verneinenden Antwort hingegen wird die Instruktion des Nein-Zweiges

Maxl nimmt ein Brett vom Waggon

ausgeführt.
Jede Frage zieht also immer eine bedingte Verzweigung nach sich, entweder zu ihrem Ja-Zweig oder zu ihrem Nein-Zweig. In welchem dieser beiden Zweige nun das Programm fortgesetzt wird, hängt von der Bedingung ab, ob die Antwort ja oder nein lautet. Als eine solche bedingte Verzweigung ist zum Beispiel der Hinweis

Programm mit der Instruktion ENDE fortführen bei Antwort ja

anzusehen. Streng genommen müßte darin auch noch der zusätzliche Vermerk erscheinen, wo das Programm bei der Antwort „nein" fortzusetzen wäre, nämlich bei der Instruktion der folgenden Zeile. Doch ein solcher Verweis auf den Nein-Zweig bleibt

einem erspart, wenn man ihn stillschweigend als selbstverständlich voraussetzt.
Eine Programmverzweigung kann aber auch unabhängig von jeder Bedingung sein, zum Beispiel

Programm mit Instruktion GEHE fortführen

In einem solchen Fall spricht man von einer unbedingten Verzweigung, der das Programm in jedem Falle folgt.
Nach allen diesen Bemerkungen läßt sich leicht verfolgen, wie das Programm nach dem Zeilen-Schema instruktionsweise Zeile für Zeile ausgeführt wird.

Name	Instruktion
BEGINN	Ist der Waggon leer?
	Programm mit Instruktion ENDE fortführen bei Antwort ja
	Maxl nimmt ein Brett vom Waggon
	Handelt es sich um ein Eichenbrett?
	Programm mit Instruktion EICHE fortführen bei Antwort ja
	Handelt es sich um ein Teakbrett?
	Programm mit Instruktion TEAK fortführen bei Antwort ja
	Handelt es sich um ein Palisanderbrett?
	Programm mit Instruktion PALISA fortführen bei Antwort ja
	Maxl bringt dieses Fichtenbrett zum Stapelplatz für Fichten-
	bretter
	Dort stapelt er dieses Fichtenbrett
GEHE	Maxl geht zum Waggon zurück
	Programm mit Instruktion BEGINN fortführen
EICHE	Maxl bringt dieses Eichenbrett zum Stapelplatz für Eichen-
	bretter
	Dort stapelt er dieses Eichenbrett
	Programm mit Instruktion GEHE fortführen
TEAK	Maxl bringt dieses Teakbrett zum Stapelplatz für Teakbretter
	Dort stapelt er dieses Teakbrett
	Programm mit Instruktion GEHE fortführen
PALISA	Maxl bringt dieses Palisanderbrett zum Stapelplatz für Palisan-
	derbretter
	Dort stapelt er dieses Palisanderbrett
	Programm mit Instruktion GEHE fortführen
ENDE	Maxl hat die Arbeit beendet.

Abb. 4

Dabei sollte aber keineswegs der Eindruck entstehen, daß Maxl's Tun und Treiben in irgendeinem Zusammenhang mit dem Computer stünde. Die Programmierung diente hier lediglich der logischen Aufgliederung eines Arbeitsablaufes in narrensichere, kleine Schritte.
Es drängt sich natürlich die Vermutung auf, daß auch der Computer irgendwelche Programme ausführen kann. Richtig, jedoch dürfen sich diese nur auf ganz spezielle Rechenvorgänge und ähnliches beziehen. Denn die Befehle, denen ein Computer zu folgen vermag, beziehen sich ausschließlich auf die Verarbeitung irgendwelcher Daten und nicht etwa auf das Abladen einer Holzfuhre.
Aber bevor auf die Arbeitsweise des Computers und dessen Programmierung eingegangen werden kann, sind noch einige vorbereitende Bemerkungen und die Behandlung weiterer Ausführungsbeispiele erforderlich.

Ausführungsbeispiel Nr. 2

In einer Röhre befindet sich eine unbekannte Anzahl weißer und schwarzer Kugeln. Am unteren Ende dieser Röhre ist eine Entnahmeschale angeordnet; sie kann jeweils eine nach außen hin sichtbare Kugel aufnehmen (Abb. 5). Wird diese Kugel entnommen, so rutscht aus der Röhre eine nachfolgende Kugel in die Entnahmeschale.
Je nach der Farbe aufeinanderfolgender Kugeln soll deren Ablage in einem der Behälter 1 bis 4 erfolgen.

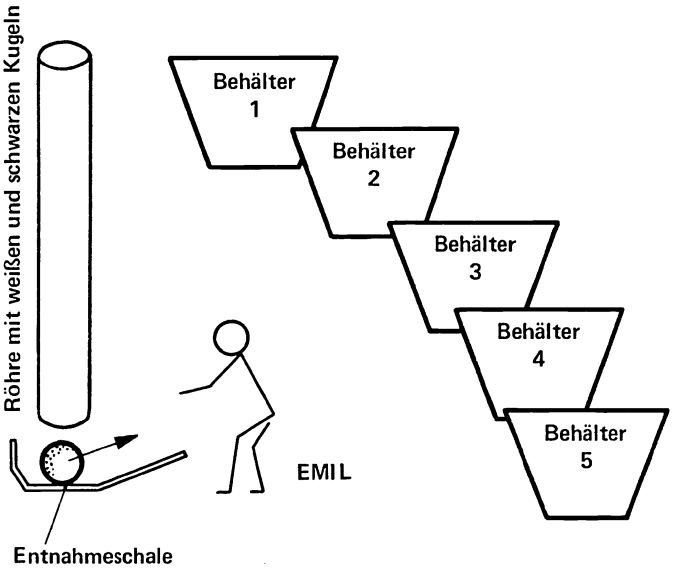


Abb. 5

Farbe der entnommenen Kugel	Farbe der in die Entnahmeschale nachfolgenden Kugel	Ablage für die entnommene Kugel
weiß	weiß	Behälter 1
weiß	schwarz	Behälter 2
schwarz	schwarz	Behälter 3
schwarz	weiß	Behälter 4

Für die allerletzte Kugel ist der Behälter 5 vorgesehen.
In Abb. 6 ist das Blockdiagramm zur Lösung dieser Aufgabe gezeigt.

Erläuterungen zu Abb. 6:

Am Anfang ist zu prüfen, ob überhaupt Kugeln in der Röhre sind. Wäre die Entnahmeschale leer, könnte Emil die geforderte Arbeit gar nicht beginnen. In einem solchen Fall wäre dafür zu sorgen, erst einmal die Röhre mit Kugeln zu füllen.
Für den weiteren Ablauf des Programmes gibt es mehrere Lösungsmöglichkeiten – eine davon ist in Abb. 6 dargestellt. Wie in allen Fällen, ist darin unbedingt auf die logisch richtige Reihenfolge der gestellten Fragen zu achten. So darf man sich zum Beispiel erst dann nach der Farbe einer Kugel in der Entnahmeschale erkundigen, wenn man sich zuvor vergewissert hat, daß die Entnahmeschale nicht etwa leer ist. (Es wäre völlig unsinnig, nach der Farbe einer nicht vorhandenen Kugel zu fragen).
Sofern die geringsten Zweifel an der Richtigkeit des Blockdiagrammes auftreten, sollte dieses einer theoretischen Überprüfung unterzogen werden.
Denn trotz aller erdenklicher Sorgfalt und Überlegung können auch einem Programmierer Fehler unterlaufen. Selbst der erfahrenste unter ihnen wird erst dann seine „Hand für das Programm ins Feuer legen", wenn er sich zuvor von dessen Richtigkeit überzeugt hat. Es wäre unverantwortlich von ihm, würde er sich diese Vergewisserung nicht verschaffen: Die Folgen wären nicht auszudenken. Es wäre uns zwar so ziemlich einerlei, ob irgendwelche Kugeln in den falschen oder richtigen Behälter gelangen, doch diese Gleichgültigkeit änderte sich schlagartig, wenn sich das Programm zum Beispiel auf die monatliche Gehaltszahlung erstreckte. Was würden Sie wohl als Angestellter empfinden, müßten Sie aufgrund eines fehlerhaften Gehaltsprogrammes unzumutbar lange auf Ihr Geld warten?
Um derartige Situationen von vornherein zu vermeiden, egal ob es sich nun um höchst oder weniger wichtige Dinge handelt, sollte jedes Programm gewissenhaft ausgetestet werden.
Selbst im alltäglichen Leben wird dem Testen (Ausprobieren) eine große Bedeutung beigemessen: Bevor ein neues Automobil auf den Markt kommt, prüft es die Herstellerfirma auf Herz und Nieren; vor einem Theaterbesuch wirft man vorsichtshalber noch einen kurzen Blick in den Spiegel, um später nicht etwa durch Goulaschreste auf der Krawatte aufzufallen.
Beim Programmieren sind die Fehlerursachen vielfältig. Diese können auf Flüchtigkeit, Vergeßlichkeit oder auf logischen Irrtümern beruhen. Um solche Fehler aufzudecken, macht man die Probe auf's Exempel. Dabei wird nachgeprüft, wie angenommene Testfälle gemäß den einzelnen Instruktionen des Blockdiagrammes verarbeitet werden. Man simuliert praktisch einen Programmablauf und vergleicht anschließend das Testergebnis mit der laut Aufgabenstellung geforderten Lösung. Nur eine entsprechende Übereinstimmung berechtigt zu der Annahme, daß das Blockdiagramm richtig aufgestellt wurde. Andernfalls müßte der Programmierer das Blockdiagramm korrigieren. Bei der Zusam-

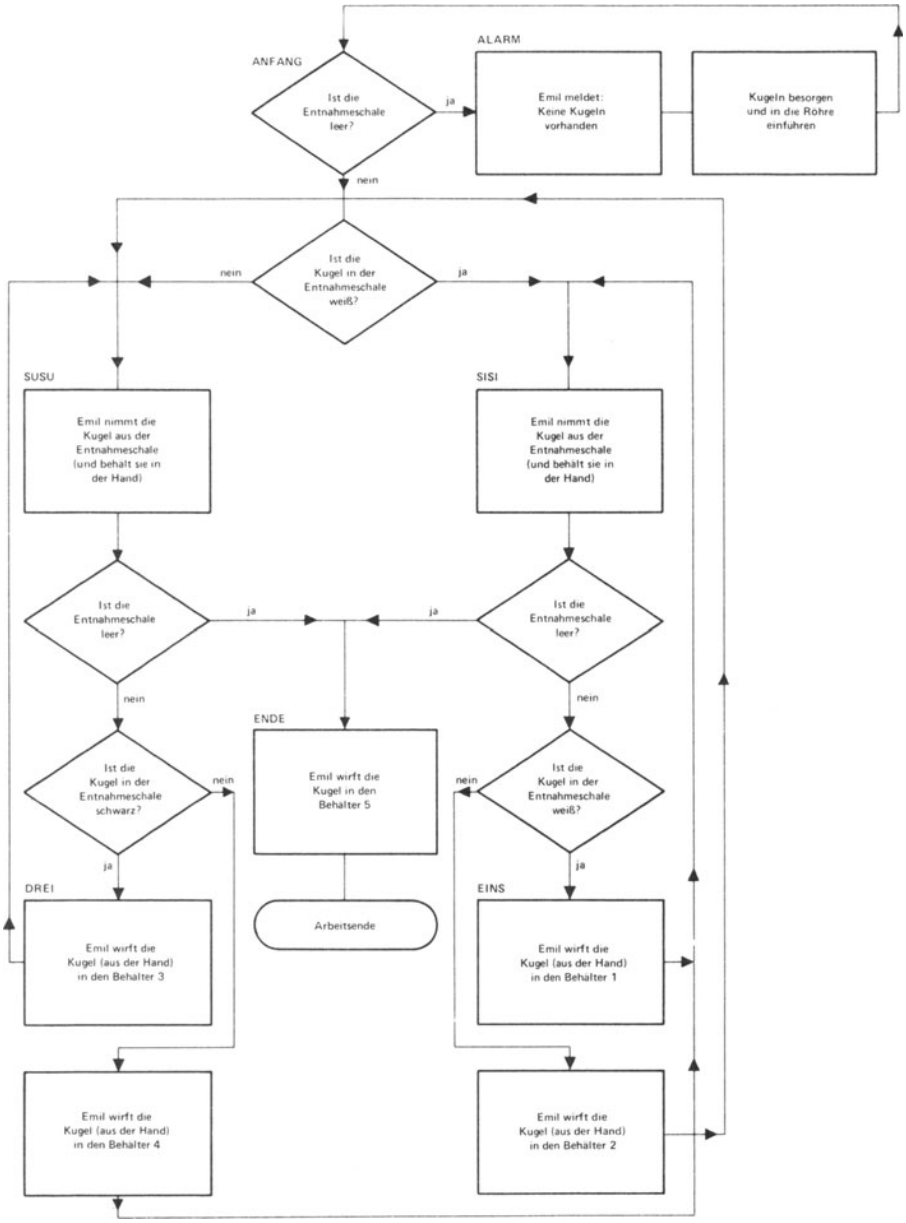
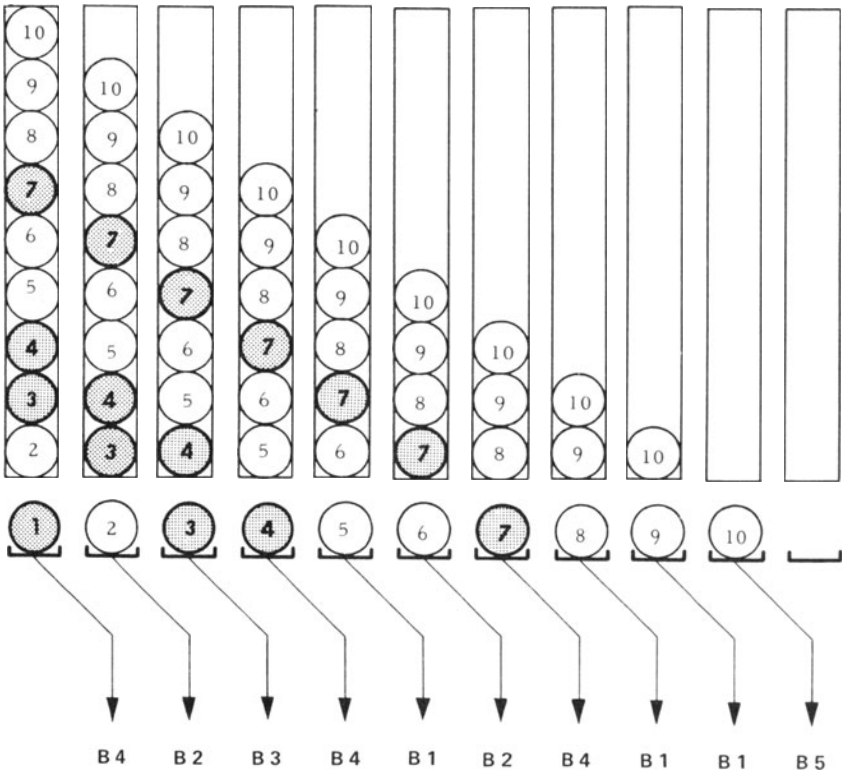

ANFANG
Ist die Entnahmeschale leer?
ja
ALARM
Emil meldet: Keine Kugeln vorhanden
Kugeln besorgen und in die Röhre einführen
nein
Ist die Kugel in der Entnahmeschale weiß?
nein
ja
SUSU
Emil nimmt die Kugel aus der Entnahmeschale (und behält sie in der Hand)
SISI
Emil nimmt die Kugel aus der Entnahmeschale (und behält sie in der Hand)
Ist die Entnahmeschale leer?
ja
ja
Ist die Entnahmeschale leer?
nein
nein
ENDE
Ist die Kugel in der Entnahmeschale schwarz?
nein
Emil wirft die Kugel in den Behälter 5
nein
Ist die Kugel in der Entnahmeschale weiß?
ja
ja
DREI
EINS
Arbeitsende
Emil wirft die Kugel (aus der Hand) in den Behälter 3
Emil wirft die Kugel (aus der Hand) in den Behälter 1
Emil wirft die Kugel (aus der Hand) in den Behälter 4
Emil wirft die Kugel (aus der Hand) in den Behälter 2

Abb. 6

menstellung der Testfälle sollte man mit sehr viel Überlegung vorgehen, um alle charakteristischen Fälle zu erfassen, die später in der Praxis irgendwann einmal auftreten könnten.
Und nun prüfen Sie bitte einmal nach, ob auch wirklich für die in Abb. 7 aufgeführten Testfälle alle Kugeln in den richtigen Behälter gelangen. Seien Sie jedoch auch noch weiterhin mißtrauisch – stellen Sie sich weitere Testfälle zusammen, einschließlich der Möglichkeit, in der Röhre wären nur Kugeln einer einzigen Farbe oder gar nur eine einzige Kugel.
Erst wenn das Blockdiagramm für richtig befunden wurde, darf das zugehörige Zeilen-Schema niedergeschrieben werden. Die Zeilen-Schemata in den Abbildungen 8, 9 und 10 beziehen sich auf das gleiche Blockdiagramm (Abb. 6), sie unterscheiden sich lediglich in der Aufeinanderfolge einzelner Programmteile; doch derartige Festlegungen bleiben Ihrer Willkür überlassen.



(B = Behälter)

Abb. 7

Ein solches Zeilen-Schema ist als eine Art Rezept aufzufassen, nach dem jeder, der es zu lesen versteht, die geforderte Arbeit verrichten kann.
Jetzt verfolgen Sie bitte noch das Ausführungsbeispiel Nr. 3 und dann haben Sie bereits die „Hürde" des Programmierens genommen.

Name	Instruktion
ANFANG	Ist die Entnahmeschale leer?
	Programm mit Instruktion ALARM fortführen bei Antwort ja
	Ist die Kugel in der Entnahmeschale weiß?
	Programm mit Instruktion SISI fortführen bei Antwort ja
SUSU	Emil nimmt die Kugel aus der Entnahmeschale
	Ist die Entnahmeschale leer?
	Programm mit Instruktion ENDE fortführen bei Antwort ja
	Ist die Kugel in der Entnahmeschale schwarz?
	Programm mit Instruktion DREI fortführen bei Antwort ja
	Emil wirft die Kugel in den Behälter IV
	Programm mit Instruktion SISI fortführen
DREI	Emil wirft die Kugel in den Behälter III
	Programm mit Instruktion SUSU fortführen
SISI	Emil nimmt die Kugel aus der Entnahmeschale
	Ist die Entnahmeschale leer?
	Programm mit Instruktion ENDE fortführen bei Antwort ja
	Ist die Kugel in der Entnahmeschale weiß?
	Programm mit Instruktion EINS fortführen bei Antwort ja
	Emil wirft die Kugel in den Behälter II
	Programm mit Instruktion SUSU fortführen
EINS	Emil wirft die Kugel in den Behälter I
	Programm mit Instruktion SISI fortführen
ALARM	Emil meldet: Keine Kugeln vorhanden
	Kugeln besorgen und in die Röhre einführen
	Programm mit Instruktion ANFANG fortführen
ENDE	Emil wirft die letzte Kugel in den Behälter V
	Arbeitsende

Abb. 8

Name	Instruktion
ANFANG	Ist die Entnahmeschale leer?
	Programm mit Instruktion ALARM fortführen bei Antwort ja
	Ist die Kugel in der Entnahmeschale weiß?
	Programm mit Instruktion SISI fortführen bei Antwort ja
SUSU	Emil nimmt die Kugel aus der Entnahmeschale
	Ist die Entnahmeschale leer?
	Programm mit Instruktion ENDE fortführen bei Antwort ja
	Ist die Kugel in der Entnahmeschale schwarz?
	Programm mit Instruktion DREI fortführen bei Antwort ja
	Emil wirft die Kugel in den Behälter IV
	Programm mit Instruktion SISI fortführen
EINS	Emil wirft die Kugel in den Behälter I
	Programm mit Instruktion SISI fortführen
DREI	Emil wirft die Kugel in den Behälter III
	Programm mit Instruktion SUSU fortführen
ALARM	Emil meldet: Keine Kugeln vorhanden
	Kugeln besorgen und in die Röhre einführen
	Programm mit Instruktion ANFANG fortführen
SISI	Emil nimmt die Kugel aus der Entnahmeschale
	Ist die Entnahmeschale leer?
	Programm mit Instruktion ENDE fortführen bei Antwort ja
	Ist die Kugel in der Entnahmeschale weiß?
	Programm mit Instruktion EINS fortführen bei Antwort ja
	Emil wirft die Kugel in den Behälter II
	Programm mit Instruktion SUSU fortführen
ENDE	Emil wirft die letzte Kugel in den Behälter V
	Arbeitsende

Abb. 9

Name	Instruktion
ANFANG	Ist die Entnahmeschale leer?
	Programm mit Instruktion ALARM fortführen bei Antwort ja
	Ist die Kugel in der Entnahmeschale weiß?
	Programm mit Instruktion SISI fortführen bei Antwort ja
SUSU	Emil nimmt die Kugel aus der Entnahmeschale
	Ist die Entnahmeschale leer?
	Programm mit Instruktion ENDE fortführen bei Antwort ja
	Ist die Kugel in der Entnahmeschale schwarz?
	Programm mit Instruktion DREI fortführen bei Antwort ja
	Emil wirft die Kugel in den Behälter IV
	Programm mit Instruktion SISI fortführen
ALARM	Emil meldet: Keine Kugeln vorhanden
	Kugeln besorgen und in die Röhre einführen
	Programm mit Instruktion ANFANG fortführen
SISI	Emil nimmt die Kugel aus der Entnahmeschale
	Ist die Entnahmeschale leer?
	Programm mit Instruktion ENDE fortführen bei Antwort ja
	Ist die Kugel in der Entnahmeschale weiß?
	Programm mit Instruktion EINS fortführen bei Antwort ja
	Emil wirft die Kugel in den Behälter II
	Programm mit Instruktion SUSU fortführen
DREI	Emil wirft die Kugel in den Behälter III
	Programm mit Instruktion SUSU fortführen
EINS	Emil wirft die Kugel in den Behälter I
	Programm mit Instruktion SISI fortführen
ENDE	Emil wirft die letzte Kugel in den Behälter V
	Arbeitsende

Abb. 10

Ausführungsbeispiel Nr. 3

Alle Wähler eines Bezirks haben ihre Stimmen für einen der beiden Kandidaten Pollux oder Plapper abgegeben. Emil übernimmt die Stimmenauswertung; er verliest nacheinander die Kandidatennamen. Moritz zählt, wie oft der Name Pollux fällt; Max tut gleiches für Plapper. Nach dem Verlesen aller Stimmzettel soll das Wahlergebnis festgestellt werden: Hat Pollux oder Plapper das „Rennen gemacht" oder ist die Wahl unentschieden ausgegangen? (Es gibt keine ungültigen Stimmen).
In Abb. 11 ist zu Ihrer Orientierung ein Blockdiagramm zur Lösung dieser Aufgabe angegeben.

Nachdem Sie jetzt einen Eindruck von der Programmierung allgemeiner Arbeitsabläufe gewonnen haben, brauchen diese Kenntnisse nur noch auf die elektronische Datenverarbeitung abgestellt zu werden. Aber bevor wir uns direkt dem Computer zuwenden können, sind noch die Lochkarten zu erwähnen, mit deren Hilfe es möglich ist, dem Computer etwas mitzuteilen: Anweisungen, Zahlen oder alphabetische Angaben.

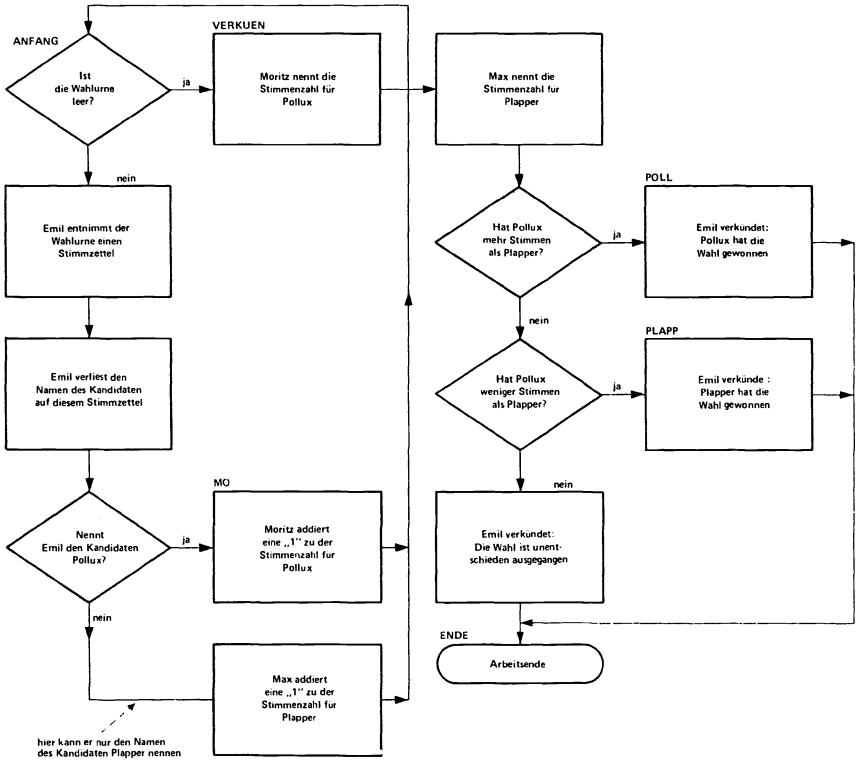


Abb. 11

2. Die Lochkarte

Bestimmt ist sie Ihnen schon einmal begegnet und vielleicht erinnern Sie sich noch an die vielen kleinen rechteckigen Löcher.
Das „Geheimnis" dieser Lochungen ist schnell geklärt; zuvor bedarf es nur noch eines kurzen Hinweises zu dem Aufbau einer Lochkarte (Abb. 12, Abb. 14):
Die Lochkarte weist insgesamt 80 Spalten auf. In jeder dieser Spalten kann ein Zeichen (Ziffer, Buchstabe, Sonderzeichen) durch eine bzw. mehrere Lochungen dargestellt werden.
Je nachdem, wo diese Lochungen angebracht sind, spricht man von einer Null-Lochung (0), einer Einer-Lochung (1), einer Zweier-Lochung (2) usw. Am oberen Lochkartenrand können außerdem noch sogenannte Elfer- und Zwölfer-Lochungen auftreten (Abb. 12).
Für die Zeichendarstellung mit Hilfe von Lochungen gibt es verschiedene Codes. Einer der bekanntesten ist hier wiedergegeben:

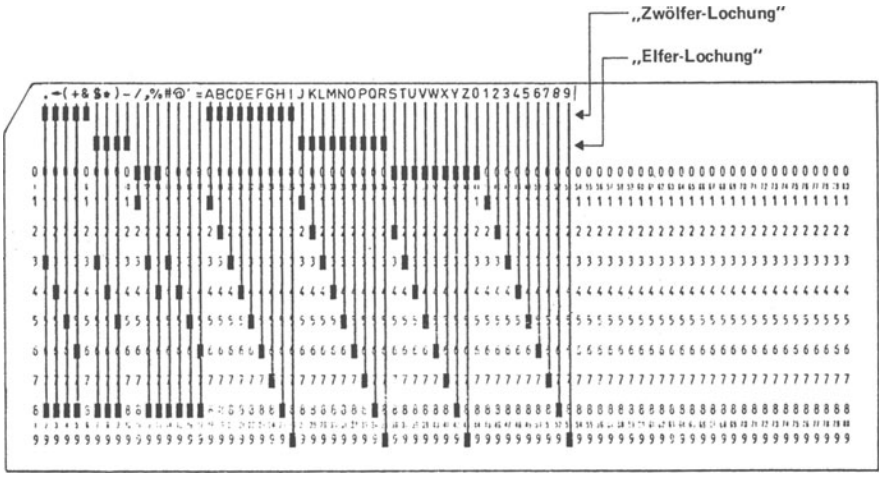


Abb. 12

Dieser Code wird den folgenden Erläuterungen auch zugrundegelegt.
Wie aus Abb. 12 hervorgeht, werden Ziffern jeweils durch eine einzige Ziffernlochung (0, 1, 2 bis 9) dargestellt.
Buchstaben hingegen erfordern immer eine ganz bestimmte Kombination zweier Lochungen, einer Ziffernlochung (1, 2 bis 9) und einer Zonenlochung (0, 11 oder 12). *)
Sonderzeichen wie ? , %, (,), etc. bestehen aus Lochkombinationen, die zur Darstellung von Buchstaben nicht benutzt werden. In Abb. 13 sind alle Zeichen mit den zugehörigen Lochungen tabellarisch aufgeführt.
Und nun verfolgen Sie bitte, wie die Adresse des Herrn Hugo Müller, wohnhaft in 8000 München, Preussen-Allee 13, in eine Lochkarte übertragen wird (Abb. 14).

* Die Lochung 0 kann sowohl eine Ziffern- als auch eine Zonenlochung sein.

Zeichen	Lochungen (Lochkarten-Code)
A	12,1
B	12,2
C	12,3
D	12,4
E	12,5
F	12,6
G	12,7
H	12,8
I	12,9
J	11,1
K	11,2
L	11,3
M	11,4
N	11,5
O	11,6
P	11,7
Q	11,8
R	11,9
S	0,2
T	0,3
U	0,4
V	0,5
W	0,6
X	0,7
Y	0,8
Z	0,9
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
.	12,8,3
(	12,8,5
+	12,8,6
&	12
!	11,8,2
$	11,8,3
*	11,8,4
)	11,8,5
;	11,8,6
–	11
/	0,1
'	0,8,3
%	0,8,4
?	0,8,7
:	8,2
#	8,3
@	8,4
'	8,5
=	8,6
(Blank = kein Zeichen)	Keine Lochung

Abb. 13

Dabei sollen für

den Nachnamen	die Spalten 1–15
den Vornamen	die Spalten 16–30
die Postleitzahl	die Spalten 31–34
den Ort	die Spalten 35–50
die Straßenbezeichnung	die Spalten 51–80

zur Verfügung stehen. Durch diese Festlegung ist die Lochkarte zugleich in einzelne Felder aufgeteilt.
Die Vermutung, daß die Lochungen etwa „mit der Schere" herzustellen wären, ist völlig unzutreffend. Selbstverständlich macht man sich auch hier die Technik zunutze und bedient sich eines Lochers (Abb. 15). Dessen Tastatur erinnert an eine Schreibmaschine. Durch Betätigung einer ausgewählten Taste werden die Lochungen für ein Zeichen automatisch herausgestanzt. Es ist lediglich darauf zu achten, daß sich immer die gewünschte Lochkartenspalte unter der Lochstation befindet.
Ein besonderer Mechanismus sorgt für einen spaltenweisen Vorschub. In Abb. 16 ist angedeutet, in welcher Reihenfolge das Ablochen vorzunehmen ist.
Dabei könnte der Fall eintreten, daß versehentlich auch einmal eine falsche Taste gedrückt wird. Ein solcher Fehler wäre schwer zu entdecken, denn wer würde sich schon die Mühe machen und alle Spaltenlochungen nachprüfen.
Viel leichter hätte man es dagegen, wenn zuvor anstelle eines einfachen Lochers ein Schreiblocher zur Verfügung gestanden hätte. Dieser druckt nämlich während des Lochvorganges gleichzeitig das betreffende Zeichen auf den Lochkartenrand (Abb. 12, Abb. 13). So läßt sich leicht kontrollieren, welche Zeichen abgelocht sind.
Diese Randbeschriftung ist für einen Computer natürlich völlig bedeutungslos. Dessen Augenmerk konzentriert sich einzig und allein auf die Lochungen, die er viel schneller erkennen kann als wir den Klartext auf dem oberen Lochkartenrand.

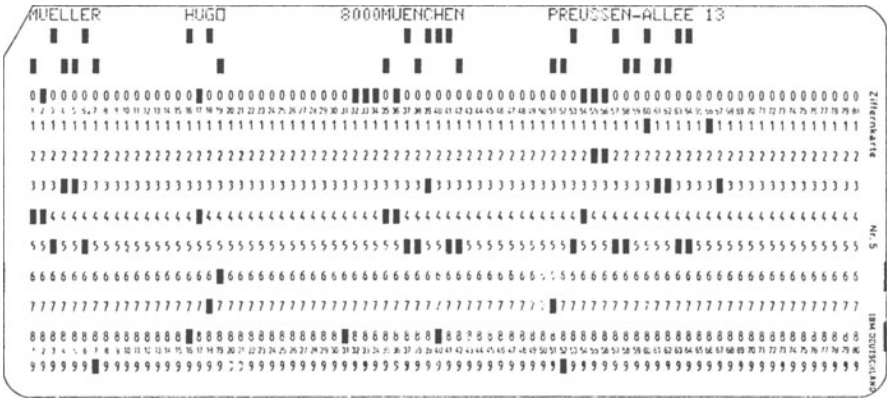


Abb. 14

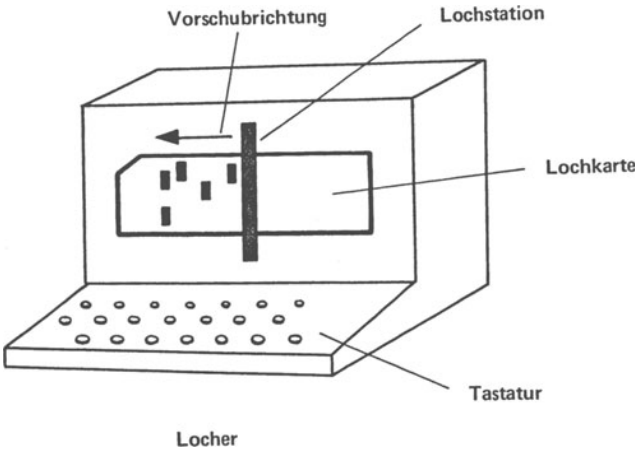


Abb. 15

Welche Spalte steht unter der Lochstation?	Welche Taste muß niedergedrückt werden?
1	M
2	U
3	E
4	L
5	L
6	E
7	R
8	Leertaste
9	Leertaste
10	Leertaste
11	Leertaste
.	.
.	.
16	H
17	U
.	.
.	.

Abb. 16

3. Ein Computer liest Lochkarten

Die Kunst, sich verständlich zu machen, ist mitunter nicht nur ein menschliches, sondern auch ein technisches Problem.
Menschen der gleichen Sprache können sich verhältnismäßig leicht untereinander etwas mitteilen. Etwas schwieriger gestaltet sich schon die Verständigung für jene, denen das Augenlicht oder die Fähigkeit zu sprechen fehlt: Taubstumme bedienen sich deshalb der Zeichensprache, Blinde der Blindenschrift (Abfühlen von Erhebungen).
Im technischen Bereich hat die Beziehung Mensch/Maschine einen besonderen Aspekt. Hier geht es im wesentlichen darum, sich irgendwie der Maschine mitzuteilen, um sie zu bestimmten Arbeiten zu bewegen.
Unser bewährtes Automobil reagiert zwar nicht auf gesprochene Worte, wohl aber auf einen Druck des Gaspedals. Einer mechanischen Registrierkasse, die addieren und das Ergebnis ausdrucken kann, wird hingegen über eine Tastatur angegeben, womit sie zu rechnen habe.
Nun wäre es durchaus denkbar, daß ein begabter Konstrukteur eine Rechenmaschine ersinnt, bei der die Zahleneingabe nicht über eine Tastatur, sondern mittels einer Lochkarte erfolgt. In diesem Falle müßte die Tastatur durch eine Abfühlstation ersetzt werden, in der die abgelochten Zahlen abzufühlen (zu lesen) wären.
Jetzt trennt uns nur noch ein kleiner Schritt vom Computer. Dessen Fähigkeiten sind natürlich bedeutend vielseitiger als die einer Registrierkasse. Er kann phantastisch schnell mit Zahlen und Buchstaben umgehen und diese Daten verarbeiten. Die damit zusammenhängenden Fragen lassen sich natürlich nicht auf einen Schlag beantworten. Da gibt es

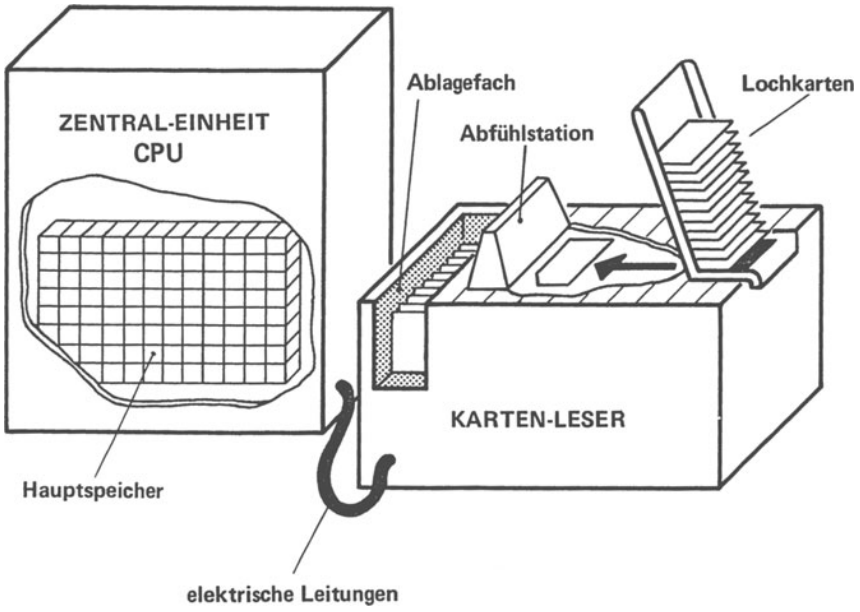


Abb. 17

zum Beispiel die Zentral-Einheit (CPU = Central Processing Unit), das „Gehirn" des Computers. Dort erfolgt die eigentliche Verarbeitung der Daten. Weitläufig wäre diese Zentral-Einheit wohl mit unserem menschlichen Gehirn vergleichbar. Das darf aber beileibe nicht zu der irrigen Annahme führen, der Computer könne denken. Die einzige Fähigkiet, die man ihm zubilligen darf, ist die enorme Schnelligkeit, mit der er bestimmte Instruktionen (Befehle) ausführt: Oftmals mehrere Millionen Rechenoperationen in der Sekunde. Die einzelnen Befehle werden jeweils in einem Programm festgelegt. Und dem Computer bleibt letztlich nichts anderes übrig, als diesen Befehlen zu folgen und die Daten wie vorgeschrieben zu verarbeiten.

In diesem Zusammenhang erhebt sich aber nun die Frage, auf welchem Wege die Lochkartendaten zur Verarbeitung in die CPU gelangen. Ein einziger Blick auf Abb. 17 genügt, um festzustellen, daß an die CPU ein Karten-Leser angeschlossen ist. In dessen Kartenmagazin liegt ein ganzer Lochkartenstapel. Sobald ein Lesebefehl erteilt wird, gelangt die zuunterst liegende Lochkarte in die Abfühlstation.

Der Abfühlvorgang kann auf verschiedene Art und Weise erfolgen. Früher wurden die Lochkarten an einem auf die Spalten ausgerichteten Satz federnder Abfühlbürsten vorbeibewegt. Beim Auftreten einer Lochung schloß die entsprechende Abfühlbürste kurzzeitig einen Kontakt und löste so einen elektrischen Impuls aus.

Heute werden die Kartenlochungen zumeist foto-elektrisch abgefühlt. Dabei sind die Abfühlbürsten durch elektrische Lämpchen ersetzt. Das von einem solchen Lämpchen ausgehende Licht gelangt nur durch eine Kartenlochung hindurch zu einem ihr gegenüber angeordneten lichtempfindlichen Element, welches daraufhin einen elektrischen Impuls abgibt.

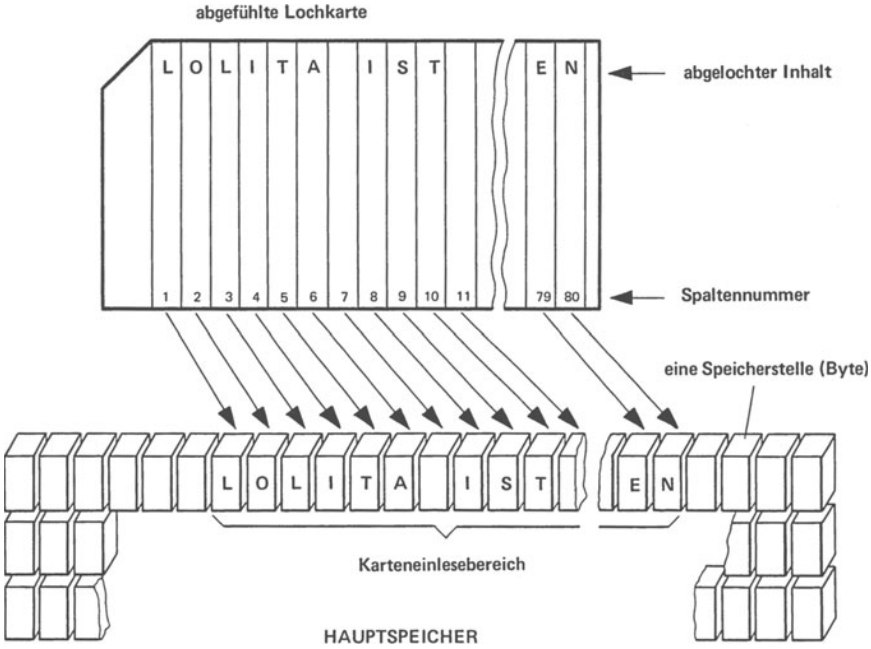


Abb. 18

Die in Impulse umgesetzten Spaltenlochungen werden sofort über elektrische Leitungen in die CPU übertragen und dort im Karteneinlesebereich des Hauptspeichers vermerkt. So erfährt der Computer, welche Zeichen in der abgefühlten (eingelesenen) Lochkarte abgelocht sind. Damit geht aber auch das Interesse des Computers an der soeben eingelesenen Lochkarte verloren, die daraufhin automatisch in das Ablagefach des Karten-Lesers gelangt.
Der Hauptspeicher bedarf noch einer ausführlicheren Erläuterung: Nicht ganz zu Unrecht nennt man ihn oft das „Computer-Gedächtnis". Diese Bezeichnung leitet sich bestimmt von der Tatsache ab, daß dort alle wichtigen Angaben gespeichert werden, die für die Ausführung eines Programmes unerläßlich sind.
Der Karteneinlesebereich ist nur ein kleiner Teil des Hauptspeichers. Er umfaßt entsprechend den 80 Spalten einer Lochkarte 80 hintereinanderliegende Speicherstellen. Jede Speicherstelle nimmt immer das Zeichen auf, welches in der zugehörigen Lochkartenspalte abgefühlt wurde (Abb. 18). Sobald alle Lochkartendaten im Karteneinlesebereich stehen, können sie auch vom Computer zu einem gewünschten Ergebnis verarbeitet werden. Nach Abschluß eines Verarbeitungszyklus steht dem Einlesen einer neuen Lochkarte nichts mehr im Wege. Wenn diese neue Lochkarte die Abfühlstation passiert, gelangt jetzt deren Inhalt in den Karteneinlesebereich. Was zuvor dort stand, interessiert ja nun nicht mehr; es wird einfach von der neuen, aktuellen Information „überschrieben".
So werden alle Lochkarten nacheinander gelesen und den Programmbefehlen entsprechend verarbeitet.

4. Der Hauptspeicher

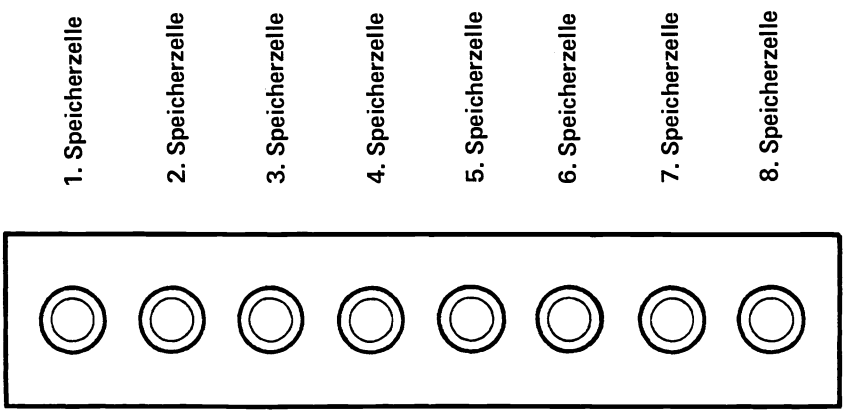
Der Hauptspeicher ist das „Gedächtnis" des Computers. Allerdings bleibt sein Erinnerungsvermögen einzig und allein auf Ziffern, Buchstaben und Sonderzeichen beschränkt. Hätte er sich beispielsweise das Wort

APOLLO

an einer bestimmten Stelle des Hauptspeichers gemerkt, so könnte er jederzeit danach gefragt werden, nach einer tausendstel Sekunde oder nach hundert Jahren, einmal oder tausendmal.
Käme man jedoch auf die Idee, APOLLO durch

ROLLER

zu ersetzen (zu überschreiben), so hilft auch kein Zureden, APOLLO wäre ein für allemal vergessen, denn nunmehr ist ROLLER aktuell.
Mit diesem Hinweis ist aber bereits etwas sehr wichtiges umschrieben worden: Der Hauptspeicher kann Information aufnehmen, speichern und beliebig oft unverändert abgeben, und alte Information kann durch neue ersetzt werden.
Jetzt ist es natürlich interessant, einiges über den technischen Aufbau des Hauptspeichers zu erfahren. Dieser ist ähnlich den Feldern eines Schachbrettes in einzelne Speicher (Abb. 18) aufgeteilt. Eine solche Speicherstelle wird vielfach Byte genannt. Jedes Byte kann jeweils ein Zeichen in einer besonders verschlüsselten Form aufnehmen und speichern. Dieser Vorgang ist technisch auf recht geniale Weise gelöst worden. Da der Computer über keine fleißigen Geisterhände verfügt, die jedes Zeichen notieren könnten, ließen sich die Entwicklungsingenieure etwas sehr Sinnvolles einfallen. Und hier ist die Lösung:



1 Speicherstelle = 1 Byte

Abb. 19

Eine Speicherstelle (Byte) wird aus acht nebeneinanderliegenden Speicherzellen gebildet. Jede Speicherzelle hat die Fähigkeit, eine „Ja-" oder „Nein-Information" aufzunehmen und speichern zu können. Eine solche Speicherzelle ist ein elektronisches Element, das entweder ein- oder ausgeschaltet ist. Diese beiden Zustände werden mit einer binären „1" oder „0" gekennzeichnet. Die Beziehung binär deutet an, daß insgesamt nur zwei unterscheidbare Zustände möglich sind: An („1") oder aus („0"); Die Speicherzelle ist also ein binäres Speicherelement.
Nun liegt bereits die Vermutung nahe, daß die einzelnen Zeichen durch entsprechende Kombinationen ein- und ausgeschalteter Speicherzellen innerhalb eines Bytes dargestellt werden können. Man spricht hierbei von einem Code. In Abb. 20 ist einer der weitverbreitesten 9-Bit-Codes, der sogenannte EBCDIC-Code, dargestellt. Als Beispiel sei die Verschlüsselung des Buchstaben E erläutert:

E 1100 0101

Für die Darstellung des E in einem Byte sind das 1., 2., 6. und 8. Bit eingeschaltet, während alle anderen Bits ausgeschaltet sind*). Dieser Sachverhalt wird, wie zuvor bereits angegeben, durch binäre Nullen und Einsen gekennzeichnet.
Der Abb. 20 ist zu entnehmen, daß ein Zeichen in einer Lochkartenspalte anders als in einem Byte dargestellt wird. Das ist durchaus verständlich, denn mit einer Lochkarte kann man nun einmal nicht so verfahren wie mit Speicherzellen.
Der Weg, der von den Zeichenlochungen zu den Speicherzellen, führt, ist voller elektronischer Eleganz. Das Ein- und Ausschalten der Speicherzellen bewirken elektrische Impulse. Diese werden beim Abfühlen der Lochungen einer Lochkarte erzeugt und gelangen über Leitungen und Codierschaltungen zum zugewiesenen Speicherplatz des Hauptspeichers.
Bei früheren Computergenerationen wurden die Speicherzellen von kleinen Magnetringkernen gebildet. Sie konnten zwei verschiedene magnetische Zustände einnehmen: sie konnten entweder ein- oder ausgeschaltet sein.
Jeder Magnetringkern war mit einer Impulsleitung durchzogen (vereinfachte Darstellung). Sobald ein elektrischer Impuls eine solche Leitung passierte, wurde der zugehörige Kern ein- oder ausgeschaltet. Das Einschalten bewirkte ein positiver, das Ausschalten ein negativer Impuls. Diese Erklärungen mögen zunächst noch etwas befremdend klingen, aber ohne elektrische Impulse und Magnetringkerne hätte es keinen Computer gegeben. Und wenn man erfährt, daß die Schaltzeit eines Kernes in der Größenordnung von nur 0,0000001 sec. lag, so vermittelt das nur einen kleinen Einblick in elektronische Geschwindigkeiten.
Durch die rasanten Fortschritte in der Halbleitertechnik konnten die Magnetringkerne bald durch sogenannte integrierte Halbleiterschaltungen (Kapitel 25) ersetzt werden. Die Speicherzellen wurden damit immer kleiner. Fast unvorstellbar, wenn wir heute vernehmen, daß auf einer Fläche von 1 cm^2 eines Halbleiterkristalls heute ca. 180 000 Speicherzellen untergebracht, integriert sein können und daß die Schaltzeiten moderner Speicherzellen ca. 0,000000001 sec. betragen. Nur so sind die extrem hohen Rechengeschwindigkeiten neuester Super-Computer mit einigen Millionen Additionen pro Sekunde zu erklären.

* Die Bezeichnungsweise, eine Speicherzelle sei ein „Bit", das ein- oder ausgeschaltet sei, hat sich vielfach eingebürgert. Das entspricht jedoch nicht der korrekten Bedeutung, wonach „bit" die Einheit der Informationsmenge ist.

Zeichen	Lochungen (Lochkarten-Code)	EBCDIC-Code
A	12,1	1100 0001
B	12,2	1100 0010
C	12,3	1100 0011
D	12,4	1100 0100
E	12,5	1100 0101
F	12,6	1100 0110
G	12,7	1100 0111
H	12,8	1100 1000
I	12,9	1100 1001
J	11,1	1101 0001
K	11,2	1101 0010
L	11,3	1101 0011
M	11,4	1101 0100
N	11,5	1101 0101
O	11,6	1101 0110
P	11,7	1101 0111
Q	11,8	1101 1000
R	11,9	1101 1001
S	0,2	1110 0010
T	0,3	1110 0011
U	0,4	1110 0100
V	0,5	1110 0101
W	0,6	1110 0110
X	0,7	1110 0111
Y	0,8	1110 1000
Z	0,9	1110 1001
0	0	0000 0000
1	1	0000 0001
2	2	0000 0010
3	3	0000 0011
4	4	0000 0100
5	5	0000 0101
6	6	0000 0110
7	7	0000 0111
8	8	0000 1000
9	9	0000 1001
.	12,8,3	0100 1011
(	12,8,5	0100 1101
+	12,8,6	0100 1110
&	12	0101 0000
!	11,8,2	0101 1010
$	11,8,3	0101 1011
*	11,8,4	0101 1100
)	11,8,5	0101 1101
;	11,8,6	0101 1110
–	11	0110 0000
/	0,1	0110 0001
,	0,8,3	0110 1011
%	0,8,4	0110 1100
?	0,8,7	0110 1111
:	8,2	0111 1010
#	8,3	0111 1011
@	8,4	0111 1100
'	8,5	0111 1101
=	8,6	0111 1110
(Blank = kein Zeichen)	Keine Lochung	0100 0000

Abb. 20

5. Der Computer erhält seine erste Arbeit

Würden Sie es für zumutbar halten, die Daten von Hunderten oder gar Tausenden von Lochkarten auf einer Liste aufschreiben zu müssen. Eine tagelange, mühsame Arbeit stünde Ihnen bevor, wenn Sie nicht auf den Gedanken kämen, sich eines Computers zu bedienen. Dieser muß allerdings Schritt für Schritt angewiesen werden, was er im einzelnen tun soll. Solche Anweisungen nimmt er nur in Form eines Programmes entgegen. Wie in einem Rezept werden darin die einzelnen „Handgriffe" festgelegt, die er später ausführen soll. Natürlich ist in dem Programm streng auf die logisch richtige Reihenfolge der einzelnen Befehle (Instruktionen) zu achten. Denn das „Elektronengehirn" kann nämlich nicht denken, sondern lediglich alle Befehle stur nacheinander befolgen. Beim Programmieren dürfen selbstverständlich nur solche Befehle verwendet werden, die der Computer auch ausführen kann, die zu seinem Befehlsrepertoire gehören. Es wäre zum Beispiel zwecklos, ihn mit dem

Scheren eines Schafes

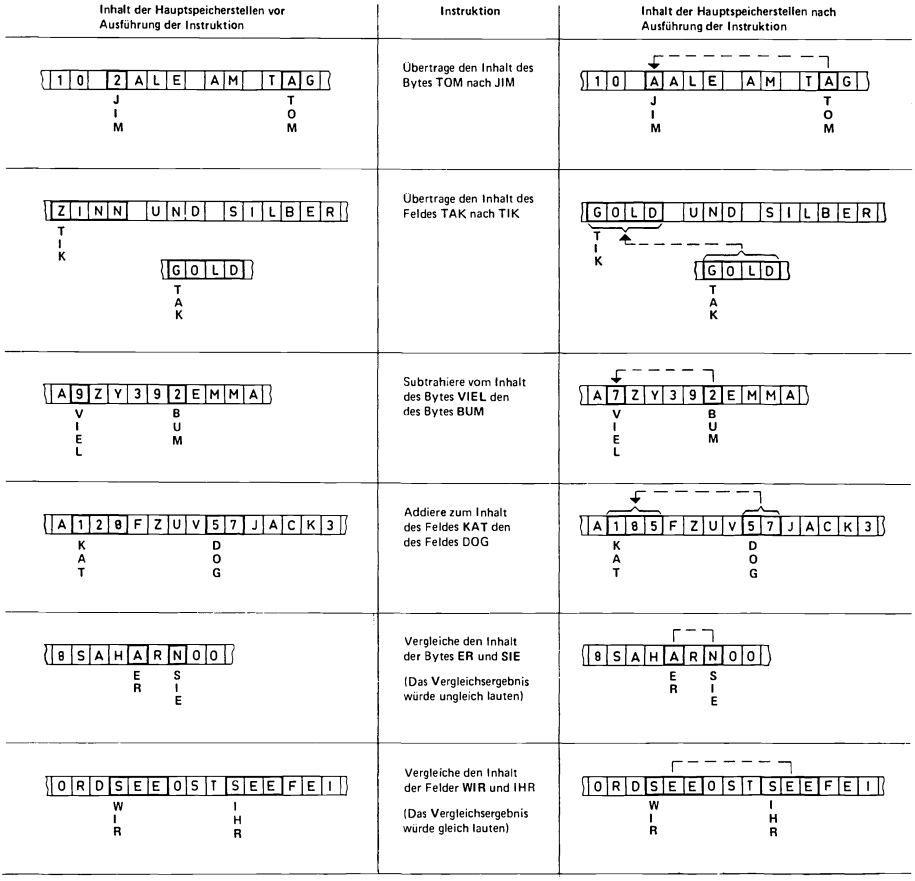
zu beauftragen und zu überfordern. Die Fähigkeiten des Computers liegen auf einem ganz anderen Sektor. So ist er imstande, in kürzester Zeit

Lochkarten zu lesen,
Zeilen zu drucken,
Rechenoperationen auszuführen,
Übertragungen und Vergleiche innerhalb des Hauptspeichers anzustellen,
Vergleichsentscheidungen zu treffen oder
die Frage zu beantworten, ob sich noch Lochkarten im Magazin des Karten-Lesers befinden.

In Abb. 21 sind verschiedene Beispiele für Rechenoperationen, Übertragungen und Vergleiche aufgeführt. Allen diesen Befehlen ist eines gemeinsam: Sie beziehen sich immer auf den Inhalt bestimmter, ausgewiesener Hauptspeicherstellen. Jedes Byte, das mit einem anderen verknüpft werden soll, ist durch einen frei wählbaren Namen (JIM, TOM usw.) gekennzeichnet (Abb. 21).
Zumeist umfassen jedoch die gespeicherten Worte bzw. Zahlen (ZINN, 128 etc.) ein ganzes Feld, das aus mehreren nebeneinanderliegenden Speicherstellen besteht. In einem solchen Fall genügt es, dieses Hauptspeicherfeld durch Benennung seines ersten Bytes auszuweisen (Abb. 21).
Wird ein Computer aufgefordert, einen Vergleich anzustellen, so erwartet man natürlich auch eine klare Entscheidung. Und der Computer hat es verhältnismäßig einfach, muß er doch jeweils nur zwei Bytes bzw. Hauptspeicherfelder bezüglich ihres Inhalts vergleichen. Die Feststellung, die er dabei macht, erstreckt sich lediglich auf die Aussage, ob die verglichenen Positionen gleichen Inhalts sind oder nicht. Die Vergleichsantwort lautet entweder „ja" oder „nein". Somit läßt sich eine Frage, zum Beispiel

Haben die Felder WIR und IHR den gleichen Inhalt?

programmtechnisch in einen Vergleich und eine bedingte Verzweigung aufgliedern (Abb. 22).



Anmerkung: Die mit einem Namen versehenen Bytes oder Hauptspeicherfelder sind stark umrandet.

Abb. 21

Ungeklärt geblieben ist aber immer noch, auf welchem Wege der Computer seine Befehle erhält. Ein „Gut-Zureden" bliebe natürlich völlig wirkungslos und so greift man auf die bewährte Verständigungsmethode per Lochkarte zurück.
Zunächst ist die Aufgabe, die einer Datenverarbeitungsanlage übertragen werden soll, in der Art eines Programmes zu formulieren. Nach den Angaben des Blockdiagrammes läßt sich ohne Schwierigkeiten das Zeilen-Schema erstellen. Jede Zeile enthält einen Befehl, der nach besonderen Vorschriften in eine Lochkarte zu übertragen (abzulochen) ist. Auf diese Weise erhält man schließlich ein Paket sogenannter Programmlochkarten (Abb. 23). Diese Programmlochkarten sind für den Computer ungefähr ebenso wichtig, wie ein Einsatzplan für einen Astronauten. Allein die strenge Einhaltung aller Befehle garantiert einen Erfolg. Und dem Computer läßt man von vornherein gar keine andere Wahl: Gnadenlos werden ihm praktisch alle Befehle „eingetrichtert". Der Karten-Leser fühlt Programmlochkarte für Programmlochkarte ab und die verschiedenen Befehle werden lückenlos nacheinander in den Hauptspeicher geladen (Abb. 23). Mit diesen

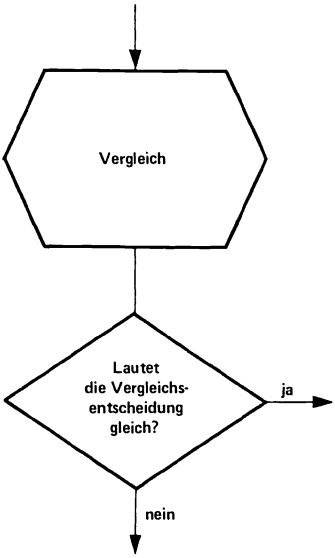


Abb. 22

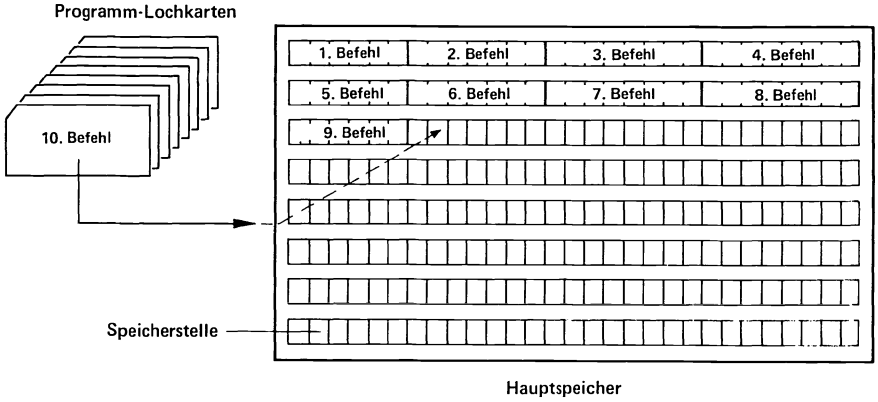


Abb. 23

Angaben weiß der Computer umzugehen. Er führt stur alle Anweisungen nacheinander aus, allerdings mit verblüffender Geschwindigkeit. Seine elektronischen Schaltkreise – diese ermöglichen überhaupt erst die Ausführung der Befehle – arbeiten phantastisch schnell. Nach Beendigung eines Programmes ist der Computer bereit, einen neuen Auftrag entgegenzunehmen. Wieder können neue Befehle in den Hauptspeicher geladen

werden – und alles, was vorher dort vermerkt war, wird überschrieben, denn jetzt ist das neue Programm aktuell.
Es mag anfangs etwas verwirren, weshalb für das Einlesen von (Daten-)Lochkarten lediglich der Karteneinlesebereich, zum Laden der Programmlochkarten hingegen der gesamte Hauptspeicher zur Verfügung steht. Im letzteren Fall werden die Befehle lückenlos aneinander gereiht (Abb. 23). Ihre Gesamtheit bildet das Programm, dessen Aufgabe es ist, Daten zu verarbeiten, mit Zahlen zu rechnen, Listen auszudrucken usw. Alle diese Daten sind dem Computer während des Programmlaufes mitzuteilen. Es ist bereits bekannt, daß dazu Lochkarten, sogenannte Daten-Lochkarten, verwendet werden können. Ausgelöst durch den Befehl

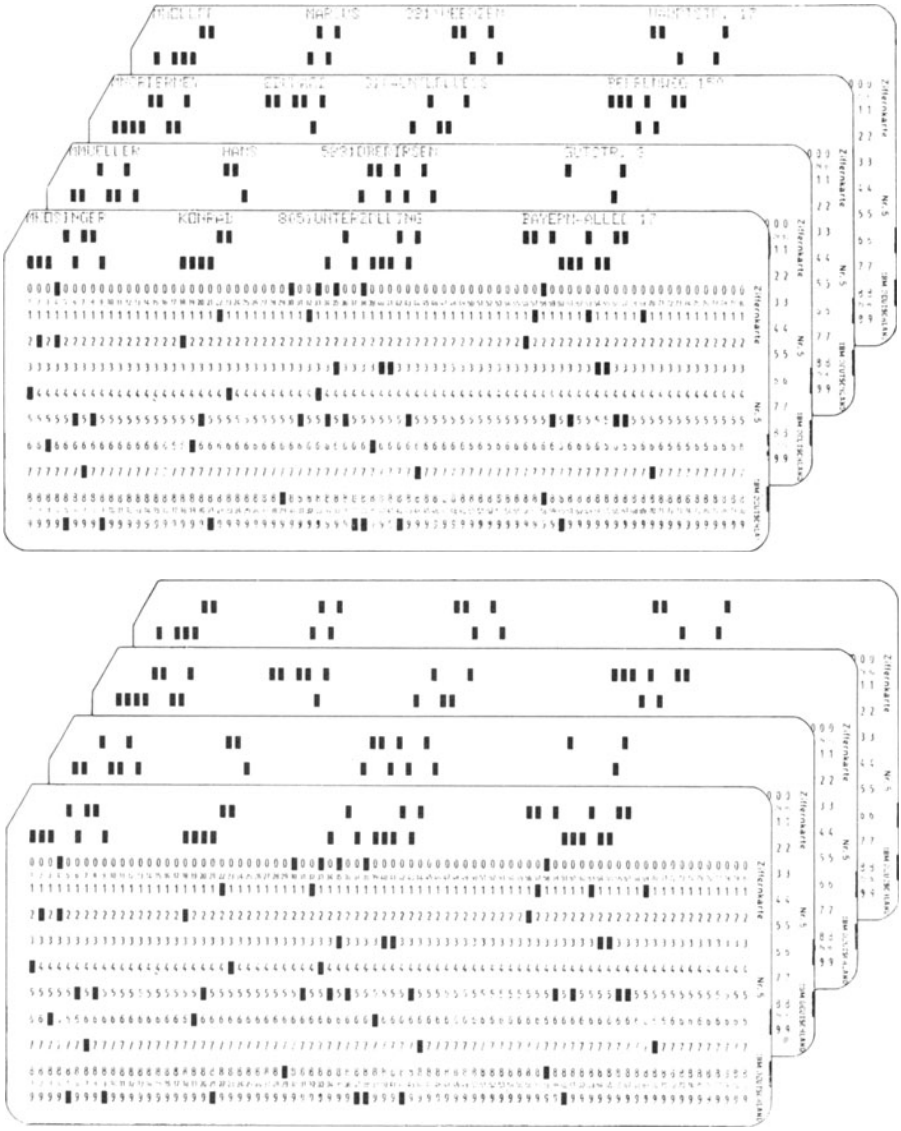
Lochkarte einlesen

erfolgt das Abfühlen einer Daten-Lochkarte im Karten-Leser und die Übermittlung deren Inhaltes (Preise, Kostenstände, Adressen usw.) in den Karteneinlesebereich. Die Größe (80 Stellen) und die Lage des Karteneinlesebereiches innerhalb des Hauptspeichers werden durch bestimmte Anweisungen im Programm festgelegt. (Nähere Erläuterungen dazu finden Sie im Ausführungsbeispiel Nr. 4). Erst wenn die Daten einer Lochkarte im Karteneinlesebereich stehen, sind sie einer Verarbeitung durch entsprechende Programmbefehle zugänglich. Zu einem späteren Zeitpunkt kann dann, wie bereits erwähnt, die nächste Daten-Lochkarte (in den Karteneinlesebereich) eingelesen und verarbeitet werden usw., bis schließlich das Magazin des Karten-Lesers leer ist.

Ausführungsbeispiel Nr. 4

Zu den einfachsten Arbeiten für einen Computer gehört das Auflisten von Lochkarten. Dabei soll der Inhalt einer jeden Lochkarte als Druckzeile auf einer Liste wiedergegeben werden. So kann man sich schnell und sicher über die abgelochten Daten informieren.
Wenn sich beispielsweise Ihr Chef für die Namen der Mitarbeiter seines Betriebes interessiert, wird er wohl kaum mit einem Berg voller Lochkarten vorlieb nehmen, in denen die Namen der Mitarbeiter abgelocht sind. Ihm dürfte es ziemlich einerlei sein, wie diese Lochkarten aussehen und ob sie eine Randbeschriftung (Abb. 24) tragen oder auch nicht. Er verlangt eine Liste und er soll sie auch bekommen:
Für diesen Zweck muß die elektronische Datenverarbeitungsanlage als Mindestausrüstung eine Zentral-Einheit, einen Karten-Leser und einen Drucker umfassen (Abb. 25). Über die Leistung solcher Einheiten ist man sicherlich nicht minder erstaunt: Manche Karten-Leser können immerhin 60 000 Lochkarten/Stunde abfühlen, manche Drucker schaffen bis zu 100 000 Zeilen/Stunde.
Es bedarf nun keiner großen Mühe mehr, das gewünschte LIST-Programm *) für die Mitarbeiter-Lochkarten aufzustellen. Alle wichtigen Schritte, die dabei zu berücksichtigen sind, wurden im Prinzip bereits erwähnt.
Vor der Erstellung des Blockdiagrammes muß man zunächst einmal sein besonderes Augenmerk auf die Feldeinteilung der Lochkarten und das gewünschte Listenbild richten. Die aufzulistenden Lochkarten (vier davon sind in Abb. 24 gezeigt) weisen folgende Feldeinteilung auf:

* Normalerweise versteht der Programmierer unter Auflisten das spaltengetreue Ausdrucken des Inhaltes einer Lochkarte in 80 Druckstellen auf der Liste (LISTEN 80/80). Im Ausführungsbeispiel Nr. 4 ist von der strengen Auslegung dieses Begriffes LISTEN abgesehen worden.



Mitarbeiter-Lochkarten mit und ohne Randbeschriftung

Abb. 24

Spalte 1	Lochung „M"
Spalte 2–17	Nachname
Spalte 18–28	Vorname
Spalte 29–32	Postleitzahl
Spalte 33–55	Ort
Spalte 56–79	Straßenbezeichnung
Spalte 80	Lochung „*" oder keine Lochung

(Die Lochung „M" in Spalte 1 ist allen Mitarbeiter-Lochkarten gemeinsam. Sie dient sozusagen zur Identifikation, denn aufgrund dieses „M" ist eine Unterscheidung zu anderen Lochkarten, die zum Beispiel Kontenstände oder Artikelbezeichnungen enthalten, möglich.) Vom LIST-Programm ist einzig und allein nur diese Karten-Art „M" zu berücksichtigen. Dabei soll allerdings eine kleine Einschränkung gelten. Verschiedene Mitarbeiter-Lochkarten weisen nämlich in Spalte 80 einen * (Stern) auf; dieser Stern deutet auf bereits aus der Firma ausgeschiedene Mitarbeiter hin. Deren Namen dürfen auf der Liste nicht mehr erscheinen. Da sich der Chef nur für die Vor- und Nachnamen der Betriebsangehörigen interessiert, braucht nur ein Teil der gesamten Lochkarte (ohne Adressangaben) ausgedruckt zu werden. Um ein besonders augenfälliges Listenbild (Abb. 26) zu erhalten, wird folgendes festgelegt:

Erste Zeile	Druckstelle 10–20	MITARBEITER
Folgende Zeilen	Druckstelle 4–19	Nachname
	Druckstelle 20–30	Vorname

Nach diesen Vereinbarungen kann endlich mit dem Aufstellen des Blockdiagrammes (Abb. 27) begonnen werden. Es mag zunächst etwas verwundern, weshalb wohl für verschiedene Blöcke eine unterschiedliche Form gewählt wurde. Doch diese Modifizierung hat sich aus Übersichtsgründen als sehr vorteilhaft erwiesen: Ein/Ausgabe-Instruktionen sind an die Form eines Parallelogrammes gebunden, Fragen stehen in einem Rhombus und Übertragungsbefehle in einfachen Rechtecken.

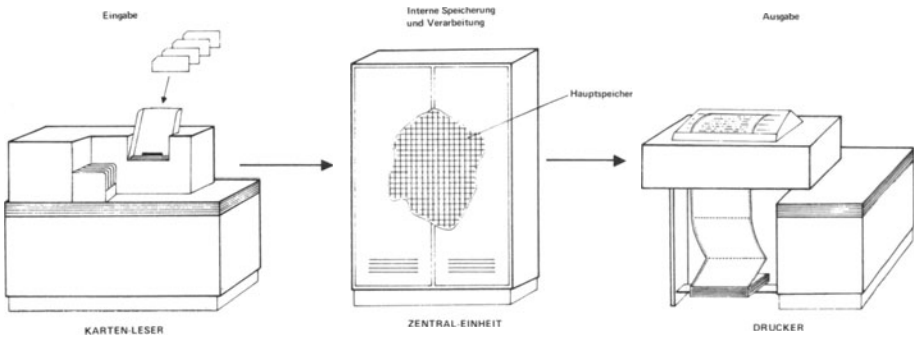


Abb. 25

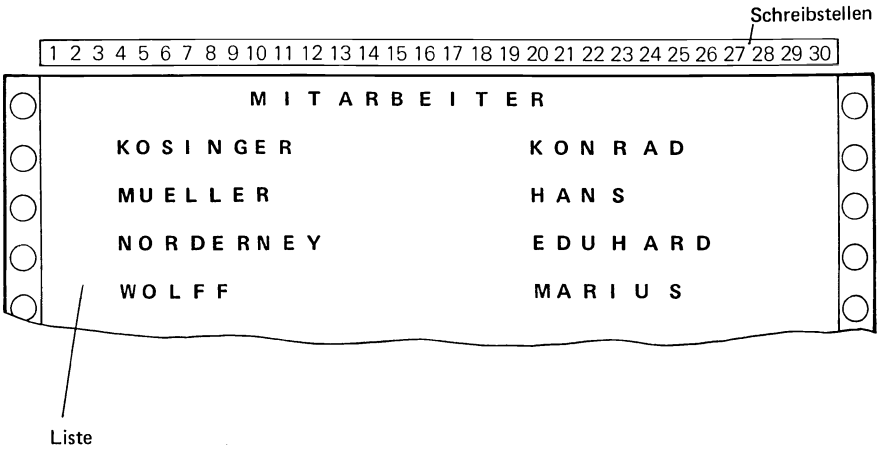


Abb. 26

Erläuterungen zu Abb. 27:

Solange sich noch Lochkarten im Kartenmagazin befinden, würde die Frage

ANFANG Sind noch Lochkarten im Karten-Magazin?

immer mit „ja" beantwortet werden. In allen diesen Fällen führt das Programm zu der Instruktion

WEITER Lies eine Lochkarte in den Karteneinlesebereich des Hauptspeichers ein

Andernfalls, also bei leerem Karten-Magazin, gäbe es keine aufzulistenden Lochkarten mehr und das Programm würde mit der Instruktion

Arbeitsende

abgeschlossen werden. Streng genommen dürfte das Karten-Magazin nur Lochkarten mit der Karten-Art „M" enthalten. Doch bei aller Aufmerksamkeit – ein Versehen ist ja nie auszuschließen – könnte sich auch einmal eine fremde Lochkarte „eingeschmuggelt" haben. Das Programm darf so etwas natürlich nicht unbemerkt quittieren. Deshalb auch die Frage

KAART Steht in der ersten Stelle des Karteneinlesebereiches ein „M"?

Sofern diese Frage ein klares „ja" nach sich zieht, ist alles in Ordnung; die gerade eingelesene Lochkarte gehört dann wirklich zur Karten-Art „M". Ein „nein" deutet allerdings einen unerwünschten „Fremdling" an, der unberücksichtigt bleiben muß – das Programm kehrt über die Instruktion namens ANFANG zum Einlesen der nächsten

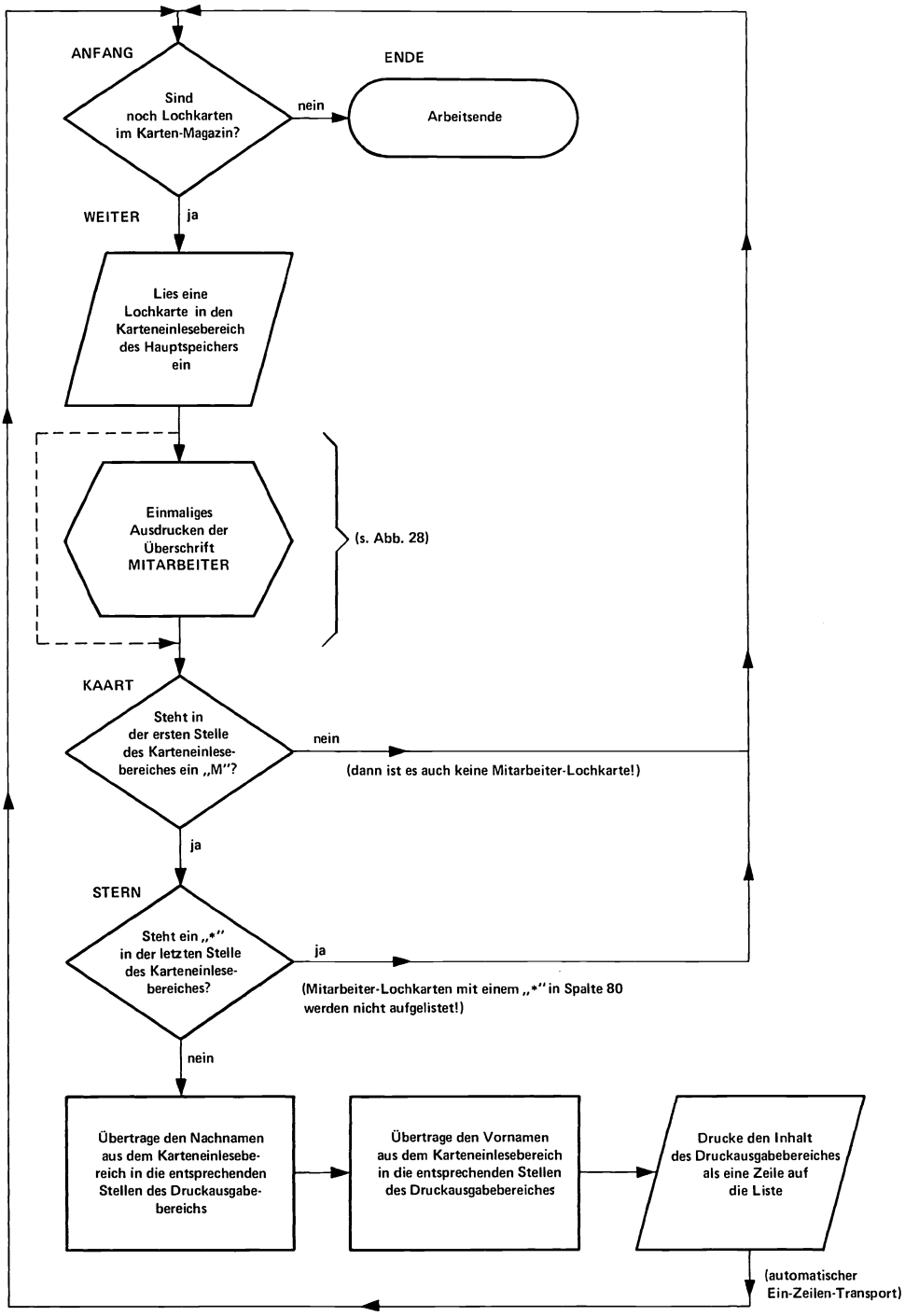

ANFANG
Sind noch Lochkarten im Karten-Magazin?
nein
ENDE
Arbeitsende
WEITER
ja
Lies eine Lochkarte in den Karteneinlesebereich des Hauptspeichers ein
Einmaliges Ausdrucken der Überschrift MITARBEITER
(s. Abb. 28)
KAART
Steht in der ersten Stelle des Karteneinlesebereiches ein „M"?
nein
(dann ist es auch keine Mitarbeiter-Lochkarte!)
ja
STERN
Steht ein „*" in der letzten Stelle des Karteneinlesebereiches?
ja
(Mitarbeiter-Lochkarten mit einem „*" in Spalte 80 werden nicht aufgelistet!)
nein
Übertrage den Nachnamen aus dem Karteneinlesebereich in die entsprechenden Stellen des Druckausgabebereichs
Übertrage den Vornamen aus dem Karteneinlesebereich in die entsprechenden Stellen des Druckausgabebereiches
Drucke den Inhalt des Druckausgabebereiches als eine Zeile auf die Liste
(automatischer Ein-Zeilen-Transport)

Abb. 27

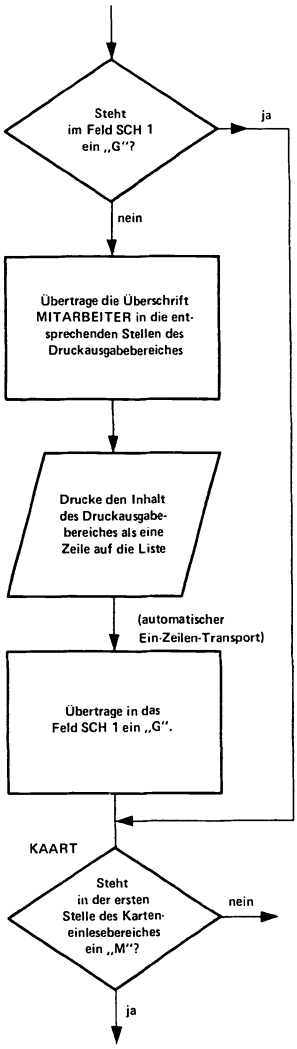


Abb. 28

Lochkarte zurück. Das daran anschließende einmalige

Ausdrucken der Überschrift „MITARBEITER"

wird an anderer Stelle erläutert.

Ein „M" in der ersten Spalte einer Lochkarte spricht zwar für die richtige Karten-Art, aber aufgelistet werden darf diese Karte deshalb immer noch nicht. Denn zunächst ist durch die Frage STERN sicherzustellen, daß sich in der letzten Stelle des Karteneinlesebereiches auch kein STERN (*) befindet. Erst dann dürfen Vor- und Nachname aus dem Karteneinlesebereich in die entsprechenden Stellen des Druckausgabebereiches übertragen und ausgedruckt werden:

Übertrage den Nachnamen aus dem Karteneinlesebereich in die entsprechenden Stellen des Druckausgabebereiches

Übertrage den Vornamen aus dem Karteneinlesebereich in die entsprechenden Stellen des Druckausgabebereiches

Drucke den Inhalt des Druckausgabebereiches als eine Zeile auf die Liste

Danach stehen die Angaben für einen Mitarbeiter auf der Liste. Das Programm kehrt wieder zu der Instruktion ANFANG zurück und nimmt seinen Fortgang.
Nach dem Ausdrucken einer jeden Zeile erfolgt ein automatischer Formularvorschub: Die Liste wird jeweils um eine Zeile unter der Druckstation vorgeschoben (Ein-Zeilen-Vorschub), so daß letzlich alle Namen der Mitarbeiter untereinander erscheinen (Abb. 26). Daß die Namen sogar alphabetisch geordnet sind, ergibt sich allerdings nicht von selbst. Der Operator, der den Computer bedient, mußte nur rechtzeitig daran denken, alle aufzulistenden Lochkarten in eine entsprechende Reihenfolge zu bringen (Kapitel 17). Jetzt ist nur noch zu klären, wie das Ausdrucken der Überschrift „MITARBEITER" im Programm berücksichtigt wird.
Zu diesem Zweck muß dieses Wort „MITARBEITER" irgendwie im Hauptspeicher verfügbar sein. Angenommen, es stünde in einem Feld namens UEBER (Abb. 29). Um es ausdrucken zu können, muß es zunächst einmal in den Druckausgabebereich übertragen werden. Nach der Erteilung des Druckbefehls erscheint es dann auf der Liste.
Im Programm ist sicherzustellen, daß die Instruktion zur Übertragung der Überschrift in den Druckausgabebereich und der daran anschließende Druckbefehl nur ein einziges Mal ausgeführt werden. Diese Forderung läßt sich leicht mit Hilfe eines sogenannten programmierten Schalters erfüllen. Das ist bei weitem kein Schalter, wie man ihn zum Beispiel bei einer elektrischen Kaffeemühle vorfindet; als programmierter Schalter läßt sich eine ganz gewöhnliche Hauptspeicherstelle (Byte) verwenden. Es bleibt völlig dem Programmierer überlassen, für ein solches Byte einen symbolischen Namen wie SCH1 festzulegen.
Der Begriff Schalter läßt vermuten, daß zwei mögliche Zustände -an bzw. aus -gekennzeichnet werden sollen. Jedem dieser beiden Zustände soll ein ganz bestimmter Inhalt des Schalterfeldes SCH1 zugrunde liegen. Man kann verabredungsgemäß festlegen, daß ein „G" (wie gesetzt) einen eingeschalteten (gesetzten) Schalter, ein „A" (wie aus) einen ausgeschalteten (nicht gesetzten) Schalter kennzeichnen soll.
Während des Programmlaufes kann dem Schalterfeld SCH1 durch eine entsprechende Übertragungsoperation jederzeit ein neuer Inhalt gegeben werden. So erfolgt zum Bei-

spiel durch Übertragung des Zeichens „G" nach SCH1 das „Setzen" des programmierten Schalters.
Außerdem ist auch jederzeit die Abfrage möglich, ob sich in SCH1 ein ganz bestimmtes Zeichen (zum Beispiel „G") befindet, ob also der Schalter gesetzt ist oder nicht.
Von diesen Möglichkeiten wird beim einmaligen Ausdrucken der Überschrift „MITARBEITER" Gebrauch gemacht. In Abb. 27 ist schematisch angedeutet, daß die Instruktion zum Ausdrucken dieser Überschrift nur ein einziges Mal durchlaufen werden darf. Kehrt das Programm zu einem späteren Zeitpunkt zum Einlesen einer neuen Lochkarte (Abb. 24) zurück, so muß anschließend die Instruktion zum Ausdrucken der Überschrift gemäß der gestrichelten Linie umgangen werden. Wie dies im einzelnen erreicht werden kann, geht aus Abb. 28 hervor:
Es sei vorausbemerkt, daß das Feld SCH1 zu Programmbeginn den per Definition (Abb. 29) zugewiesenen Inhalt „A" hat. Gelangt man während des Programmes zum allerersten Mal zu der Frage

Steht im Feld SCH1 ein „G"?

so muß die Antwort zwangsweise „nein" lauten. Folglich gelangen dann auch die Befehle

Übertrage die Überschrift „MITARBEITER" in die entsprechenden Stellen des Druckausgabebereiches

Drucke den Inhalt des Druckausgabebereiches auf die Liste

Übertrage in das Feld SCH1 ein „G"

zur Ausführung. Der weitere Verlauf des Programmes erfolgt so, wie es in Abb. 27 angedeutet ist. Dabei kehrt man natürlich zu einem späteren Zeitpunkt wieder zu der Frage (Abb. 28)

Steht im Feld SCH1 ein „G"?

zurück. Diesmal muß die Antwort allerdings „ja" lauten, denn schließlich wurde ja einige Schritte zuvor nach SCH1 ein „G" übertragen. Da der Ja-Zweig dieser Frage die Befehle zum Ausdrucken der Überschrift umgeht, wird das Wort „MITARBEITER," wie gefordert, auch kein zweites Mal auf die Liste ausgegeben.
Nachdem jetzt das vollständige Blockdiagramm vorliegt, kann das zugehörige Zeilen-Schema (Abb. 29) erstellt werden. Jede Zeile umfaßt eine Instruktion. Für die im Programm erwähnten Hauptspeicherfelder (Karteneinlesebereich, Druckausgabebereich, Schalterfeld SCH1, Überschriftsfeld UEBER) sind zusätzliche Angaben zu machen. Dem Computer ist nämlich für jedes Feld der symbolische Name, die Größe (Anzahl der Bytes) und der Inhalt mitzuteilen. (Die symbolischen Namen sind frei wählbar. Der anfangs zugewiesene Inhalt kann sich natürlich während des Programmes durch entsprechende Instruktionen verändern). Alle diese Angaben gehören zum Programm. Sie sind im Anschluß an die eigentlichen Befehle unter der Rubrik DEFINITIONEN aufzuführen (Abb. 29). Durch konsequente Benutzung der symbolischen Feldnamen können umständliche Beschreibungen der Hauptspeicherfelder, wie Karteneinlesebereich etc., entfallen. So heißt es zum Beispiel nicht mehr wie in Abb. 29

Lies eine Lochkarte in den Karteneinlesebereich des Hauptspeichers ein

sondern kurz und bündig wie in Abb. 30

Lies Lochkarte nach KARBER

(KARBER ist als Karteneinlesebereich definiert). Ein expliziter Hinweis, mit welcher Hauptspeicherstelle ein symbolisch benanntes Feld beginnt, ist nicht erforderlich. Der Computer merkt sich beim Laden des Programmes von selbst, daß zum Beispiel das Feld SCH1 auf das 381. Byte (angenommener Wert) zu liegen kommt. Sobald sich ein Programmbefehl auf SCH1 bezieht, erinnert sich der Computer an diese Adresse und stellt sich darauf ein.
Nach diesen Vorbemerkungen kann man leicht verfolgen, wie der Computer das LIST-Programm ausführt. Die Angaben des Zeilen-Schemas sollen bereits in Programmlochkarten abgelocht und in den Hauptspeicher geladen worden sein (Abb. 31). Lückenlos reiht sich dort Befehl an Befehl, gefolgt von den definierten Hauptspeicherfeldern. Die Darstellung in Abb. 31 ist natürlich schematisiert. Sie dient lediglich der Anschaulichkeit *). Aber so kann man am besten Schritt für Schritt die Arbeitsweise des Computers verfolgen: Dazu sei angenommen, im Magazin des Karten-Lesers befänden sich die in Abb. 24 gezeigten Lochkarten. Die Programmbefehle im Hauptspeicher werden nacheinander ausgeführt: Zunächst wird geprüft, ob sich überhaupt eine Lochkarte im Karten-Magazin befindet (andernfalls wäre ja das Kartenmagazin leer). Erst dann kann die Lochkarte durch die Instruktion WEITER nach KARBER eingelesen werden usw., usw. Anhand der in Abb. 24 gezeigten Eingabedaten ist es möglich, durch schrittweises Verfolgen des Programmverlaufes den Inhalt der einzelnen Hauptspeicherfelder zu bestimmten Zeitpunkten zu ermitteln (Abb. 32). Auf diese Art und Weise erwirbt man sich am besten ein echtes Gefühl für den Computer und dessen Arbeitsweise.

* In Wirklichkeit sind nämlich alle Computer-Instruktionen verschlüsselt (siehe Kap. 15).

Name	Instruktion
ANFANG	Sind noch Lochkarten im Karten-Magazin?
	Verzweige nach WEITER bei Antwort ja
ENDE	Arbeitsende
WEITER	Lies eine Lochkarte in den Karteneinlesebereich des Hauptspeichers ein
	Steht im Feld SCH 1 ein „G"?
	Verzweige nach KAART bei Antwort ja
	Übertrage die Überschrift MITARBEITER in den Druckausgabebereich
	Drucke den Inhalt des Druckausgabebereiches als eine Zeile auf die Liste
	Übertrage in das Feld SCH 1 ein „G"
KAART	Steht in der ersten Stelle des Karteneinlesebereiches ein „M"?
	Verzweige nach STERN bei Antwort ja
	Verzweige nach ANFANG
STERN	Steht ein „*" in der letzten Stelle des Karteneinlesebereiches?
	Verzweige nach ANFANG bei Antwort ja
	Übertrage den Nachnamen aus dem Karteneinlesebereich in die entsprechenden Stellen des Druckausgabebereiches
	Übertrage den Vornamen aus dem Karteneinlesebereich in die entsprechenden Stellen des Druckausgabebereiches
	Drucke den Inhalt des Druckausgabebereiches als eine Zeile auf die Liste
	Verzweige nach ANFANG

Definitonen:

	symbol. Name	Feldgröße	Inhalt
Karteneinlesebereich	KARBER	80 Bytes	
Druckausgabebereich	AUSGAB	30 Bytes	
Schalterfeld	SCH 1	1 Byte	A
Überschriftsfeld	UEBER	11 Bytes	M I T A R B E I T E R

1 2 3 4 5 6 7 8 9 10 11

30

Abb. 29

Name	Instruktion
ANFANG	Sind noch Lochkarten im Karten-Magazin?
	Verzweige nach WEITER bei Antwort ja
ENDE	Arbeitsende
WEITER	Lies Lochkarte nach KARBER
	Steht in SCH 1 „G"?
	Verzweige nach KAART bei Antwort ja
	Übertrage UEBER nach AUSGAB
	Drucke den Inhalt von AUSGAB auf die Liste
	Übertrage nach SCH 1 ein „G"
KAART	Steht in der ersten Stelle von KARBER „M"?
	Verzweige nach STERN bei Antwort ja
	Verzweige nach ANFANG
STERN	Steht ein „*" in der letzten Stelle von KARBER
	Verzweige nach ANFANG bei Antwort ja
	Übertrage den Nachnamen aus KARBER in die entsprechenden Stellen nach AUSGAB
	Übertrage den Vornamen aus KARBER in die entsprechenden Stellen nach AUSGAB
	Drucke den Inhalt von AUSGAB auf die Liste
	Verzweige nach ANFANG

Definitionen:

	symbol. Name	Feldgröße	Inhalt
Karteneinlesebereich	KARBER	80 Bytes	Blank (ohne Inhalt)
Druckausgabebereich	AUSGAB	30 Bytes	Blank (ohne Inhalt)
Schalterfeld	SCH 1	1 Byte	A
Überschriftsfeld	UEBER	11 Bytes	MITARBEITER

Abb. 30

ANFANG		ENDE
Lochkarte in Karten-Magazin?	Verzweige nach WEITER bei ja	Arbeitsende
WEITER		
Lies Lochkarte nach KARBER	Steht in SCH1 „G"?	Verzweige nach KAART bei ja
Übertrage UEBER nach AUSGAB	Drucke Inhalt von AUSGAB	Übertrage nach SCH1 „G"
KAART		
Steht in 1. Stelle von KARBER „M"?	Verzweige nach STERN bei ja	Verzweige nach ANFANG
STERN		
Steht „*" in letzter Stelle von KARBER?	Verzweige nach ANFANG bei ja	Übertrage Nachname aus KARBER nach AUSGAB
Übertrage Vorname aus KARBER nach AUSGAB	Drucke Inhalt von AUSGAB	Verzweige nach ANFANG

KARBER

Karteneinlesebereich (80 Bytes)

AUSGAB

SCH1 UEBER

A M I T A R B E I T E R

Druckausgabebereich (30 Bytes)

Abb. 31

a) Speicherauszug vor Ausführung der Instruktion KAART nach dem Einlesen der ersten Mitarbeiter-Lochkarte

KARBER

| | | |

6. Adressierung im Hauptspeicher

Das Programmieren hat sich bisher recht anschaulich und übersichtlich gestaltet. Die Verwendung symbolischer Bezeichnungen für Instruktionsnamen und Hauptspeicherfelder ist fast schon zu einer Selbstverständlichkeit geworden.
Und die prinzipielle Arbeitsweise des Computers ist bereits auch bekannt: Das Programm wird in den Hauptspeicher geladen und schrittweise, Befehl für Befehl, ausgeführt (Abb. 31). Dabei erhebt sich natürlich die Frage, wie der Computer mit symbolischen Angaben umgeht.
Nach der schematischen Darstellung in Abb. 31 scheint zunächst alles reibungslos zu verlaufen. Und dennoch ergeben sich bei genauer Überlegung einige Schwierigkeiten:
Angenommen, das Programm sei bis zu der Frage

Steht in SCH1 ein „G"?

fortgeschritten. Bei einer bejahenden Antwort müßte es nach Ausführung der folgenden Instruktion

Verzweige nach KAART bei Antwort ja

bei KAART fortgesetzt werden.
Doch woher sollte der Computer plötzlich auf Anhieb wissen, an welcher Stelle des Hauptspeichers sich diese KAART-Instruktion befindet. Es wäre immerhin denkbar, sie irgendwo, entweder am Anfang, in der Mitte oder am Ende des Hauptspeichers zu verthuten. Unserem flinken Auge fällt es natürlich nicht schwer, diese Stelle in Abb. 31 auszumachen. Dem Computer hingegen bleibt eine solche visuelle Hilfe versagt. Er muß auf elektronischem Wege herausfinden, wo die KAART-Instruktion steht. Um sie aufzuspüren, müßte eigentlich der gesamte Hauptspeicher schrittweise danach durchsucht werden – fürwahr, ein unzumutbares, zeitaufwendiges Unterfangen.
Glücklicherweise ist der Computer so konstruiert, daß von einem solchen umständlichen Suchen abgesehen werden kann. Dazu ist jeder Hauptspeicherstelle (Byte) eine fortlaufende Adresse zugeordnet (Abb. 33a). Wird der Computer zum Beispiel auf das Byte Nr. 136 verwiesen, so kann er sich, wie bei einem gezielten Telefonanruf sofort darauf einstellen.

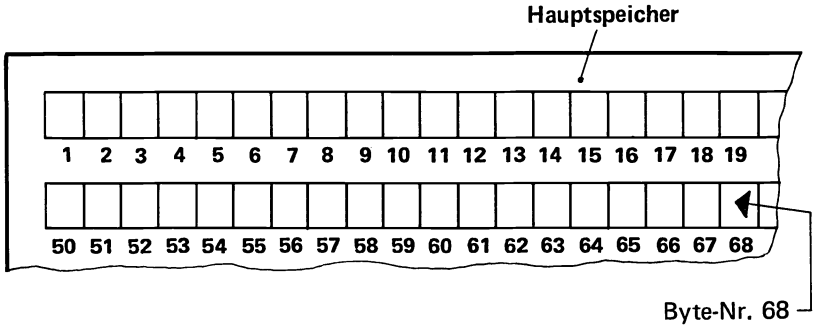


Abb. 33a

Lochkarte in Karten-Magazin? (1)	Verzweige nach Nr. 46 bei ja (16)	Arbeitsende (31)
Lies Lochkarte nach Nr. 271 bis Nr. 350 (46)	Steht in Nr. 381 ein „G"? (61)	Verzweige nach Nr. 136 bei ja (76)
Übertrage von Nr. 382 bis Nr. 392 nach Nr. 360 bis Nr. 370 (91)	Drucke Inhalt von Nr. 351 bis Nr. 380 (106)	Übertrage nach Nr. 381 ein „G" (121)
Steht in Nr. 271 ein „M"? (136)	Verzweige nach Nr. 181 bei ja (151)	Verzweige nach Nr. 1 (166)
Steht ein „*" in Nr. 350? (181)	Verzweige nach Nr. 1 bei ja (196)	Übertrage von Nr. 272 bis Nr. 287 nach Nr. 354 bis Nr. 369 (211)
Übertrage von Nr. 288 bis Nr. 298 nach Nr. 370 bis Nr. 380 (226)	Drucke Inhalt von Nr. 351 bis Nr. 380 (241)	Verzweige nach Nr. 1 (256)

271 — 286 — Karteneinlesebereich (80 Bytes) — 301

316 — 350 — 351

A M I T A R B E I T E R

Druckausgabebereich (30 Bytes) — 381 382 — 392

Abb. 33b

Stünde anstelle von

Verzweige nach KAART bei ja	(siehe Abb. 31)
Verzweige nach Nr. 136 bei ja	(siehe Abb. 33b)

so machte der direkte Hinweis auf die (absolute) Hauptspeicheradresse 136 jegliches Suchen nach der KAART-Instruktion überflüssig.
Es ist also sehr zweckmäßig, wenn sich der Computer bei Ausführung eines Programmbefehles nicht (wie in Abb. 31 vereinfach angenommen wurde) auf symbolische Bezeichnungen, sondern auf absolute Hauptspeicheradressen (Abb. 33b) beziehen kann. Darin liegt einzig und allein das ganze Geheimnis der Orientierung im Hauptspeicher.
Dessen ungeachtet verwendet aber der Programmierer in seinem Programm (Abb. 30) stets nur symbolische Bezeichnungen (ANFANG, KARBER, SCH1 etc.). Das ist einfach und übersichtlich. Aber wer wandelt nun diese symbolische Bezeichnungen in die entsprechenden Hauptspeicheradressen um? Die Antwort fällt sehr befriedigend aus: Der Computer erledigt diese Arbeit selbsttätig (siehe Kapitel 9). Dabei werden Instruktions- und Feldnamen, wie KARBER, AUSGAB etc., durch absolute Adressen ersetzt. So lautet zum Beispiel die Instruktion

Übertrage UEBER nach AUSGAB	(Abb. 31)

nach einer solche Umwandlung

Übertrage von Nr. 382 bis Nr. 392 nach Nr. 360 bis Nr. 370	(Abb. 33b)

Das Feld UEBER beginnt bei der Hauptspeicheradresse 382, das Feld AUSGABE bei 351. Nach dem Listenbild in Abb. 26 ist es erwünscht, die Überschrift MITARBEITER ab der 10. Schreibstelle erscheinen zu lassen. Dieser 10. Schreibstelle entspricht jedoch die Adresse 360 des Druckausgabebereiches AUSGAB.
Wenn man zur Vereinfachung der Darstellung einmal annimmt, jede Instruktion würde im Hauptspeicher 15 Bytes belegen, so hätte das LIST-Programm nach Abb. 30 in Adressverschlüsselung folgendes Aussehen (Abb. 33b).
Es läßt sich leicht nachprüfen, unter welchen Adressen die einzelnen Instruktionen und die definierten Bereiche anzutreffen sind. Die 18. Instruktion

Verzweige nach ANFANG

endet bei der Adresse 270 (18 x 15 = 270), so daß der daran anschließende Karteneinlesebereich KARBER (Feldgröße: 80 Bytes) von der Adresse 271–350 reicht.
An dieser Stelle sei bereits angedeutet, daß im Hauptspeicher mehr als nur die symbolischen Bezeichnungen eines Programmes verschlüsselt sind (siehe Kapitel 15). (Genau genommen müßten auch die Instruktionen in Abb. 33b zeichenweise in den einzelnen Bytes stehen. Von einer solchen Forderung wird jedoch hier noch aus Gründen der Vereinfachung abgesehen).
Doch zurück zur Tätigkeit des Programmierers. Oftmals muß sich dieser auf bestimmte Stellen eines symbolisch benannten Feldes beziehen. Möchte er zum Beispiel wissen, ob in der letzten Stelle des 80 Byte großen Karteneinlesebereiches KARBER ein „*" steht, so darf er seine Frage wie folgt formulieren:

Enthält KARBER+79 einen „*"?

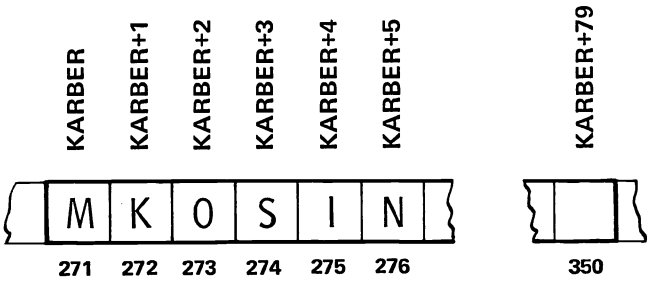
Die Bedeutung des Ausdruckes KARBER+79 ist schnell erklärt: Das Hauptspeicherfeld KARBER trägt linksbündig seinen Namen (Abb. 34). Damit ist zugleich auch dessen erste Stelle gekennzeichnet. Besteht der Wunsch, sich auf die 2. Stelle dieses Feldes zu beziehen, so genügt der Hinweis KARBER+1, die 3. Stelle wäre mit KARBER+2, die 80. und letzte Stelle mit KARBER+79 zu benennen. Alle diese Angaben sind relativ, sie beziehen sich immer auf den Feldanfang. Man spricht in diesem Zusammenhang auch von einer symbolischen relativen Adressrechnung. Sie kommt dem Programmierer sehr gelegen: Denn so kann er auf die Verwendung absoluter Hauptspeicheradressen, die er von vornherein ohnehin nicht kennt, verzichten. Die relative Adressrechnung ermöglicht es, in den einzelnen Instruktionen präzise und detaillierte Angaben machen zu können. Der noch ziemlich pauschale Verweis

> Übertrage den Vornamen aus KARBER in die entsprechenden Stellen nach AUSGAB

würde in exakterer Schreibweise

> Übertrage (in der Länge von 11 Bytes) von KARBER+17 nach AUSGAB+19

lauten. Derartige Feinheiten finden bereits im Kapitel 8 ihre Berücksichtigung. Dort erhält das LIST-Programm (Abb. 30) seinen letzten Schliff.



Relative Adressierung

Abb. 34

7. Dateien

Wenn man mit einigen Worten die Fähigkeiten eines Computers umschreiben müßte, so könnte man sich auf folgende Aussage beschränken: Der Computer kann Daten einlesen, verarbeiten und ausgeben.
Bisher wurden die Ein/Ausgabe-Befehle in recht anschaulicher Art und Weise beschrieben, zum Beispiel:

Lies eine Lochkarte in den Karteneinlesebereich des Hauptspeichers ein

oder

Drucke den Inhalt des Druckausgabebereiches des Hauptspeichers als Zeile auf die Liste.

In der symbolischen Programmierweise sind derartige umständliche Hinweise nicht mehr angebracht. Man begnügt sich deshalb mit einer kurzen, formell vorgeschriebenen Ausdrucksweise.
Danach lassen sich die soeben erwähnten Instruktionen beispielsweise zu

```
GET    KART
```

oder

```
PUT    DRUCK
```

vereinfachen. Die Operationsbezeichnungen GET und PUT sind feststehende Ausdrücke. Sie deuten entweder das Einlesen oder die Ausgabe von Daten an. Dieser Hinweis allein genügt jedoch nicht; schließlich muß über die Art der Daten auch noch etwas ausgesagt werden. Dem Computer ist dabei eindeutig vorzuschreiben, ob es sich nun um Lochkarten, Listen oder etwas anderes handelt und welchen Ein/Ausgabe(Input/Output)-Einheiten (Karten-Leser, Drucker, Karten-Stanzer usw.) diese Datenträger zuzuordnen sind. Der Programmierer kann dieser Forderung verhältnismäßig leicht nachkommen, indem er in den sogenannten Dateibeschreibungen entsprechende Angaben macht. Diese Dateibeschreibungen sind ein Teil des Programmes; sie werden dem Zeilen-Schema vorangestellt.
Der Name Datei mag zunächst noch etwas befremdend klingen. Er deutet nichts anderes als den zusammenfassenden Begriff für gleichartige Daten an. So sind zum Beispiel alle Lochkarten, die über den Karten-Leser einer Programmverarbeitung zugeführt werden, als eine Datei anzufassen. Auf einer Liste auszudruckende Daten hingegen stellen wieder eine andere Datei dar.
Um zwischen den verschiedenen Dateien unterscheiden zu können, soll jede Datei einen symbolischen Datei-Namen erhalten. Für die Lochkarten wählte der Programmierer den Namen KART, für die List-Daten den Namen DRUCK.
Jetzt läßt sich die Instruktion

```
GET    KART
```

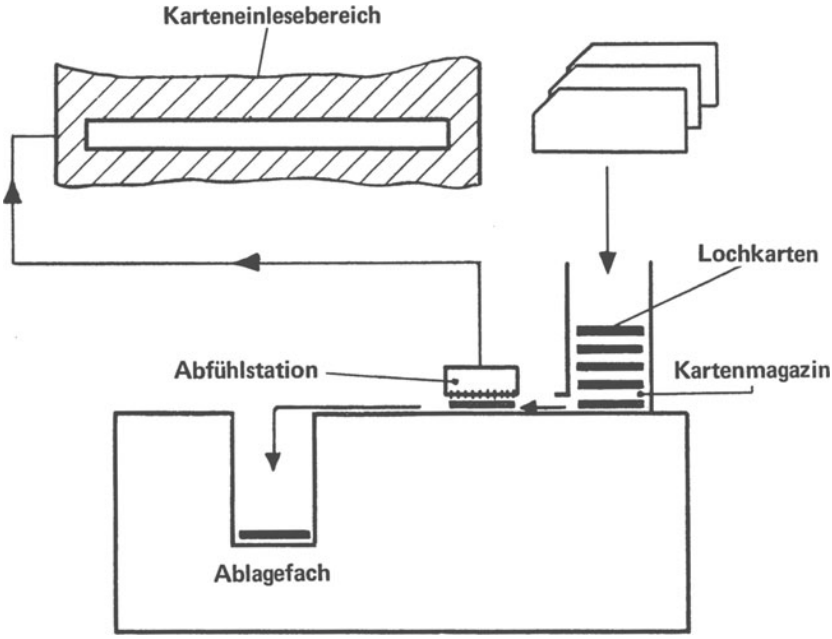


Abb. 35

schon eher verstehen. Bei deren Ausführung werden folgende Vorgänge ausgelöst (Abb. 35):

1. Automatischer Transport der im Karten-Magazin zuunterst liegenden Lochkarte in die Abfühlstation

2. Übertragung des Inhaltes dieser abgefühlten Lochkarte in den Karteneinlesebereich des Hauptspeichers

3. Ablage dieser Lochkarte im Ablagefach

Die Auswirkung der Instruktion

PUT DRUCK

Ist in Abb. 36 schematisch dargestellt:

1. Der Inhalt des Druckausgabebereiches des Hauptspeichers wird als Klartext-Zeile auf die Liste gedruckt

2. Die Liste wird automatisch um Zeilenbreite unter der Druckstation fortbewegt (wichtig für das spätere Ausdrucken einer neuen Zeile).

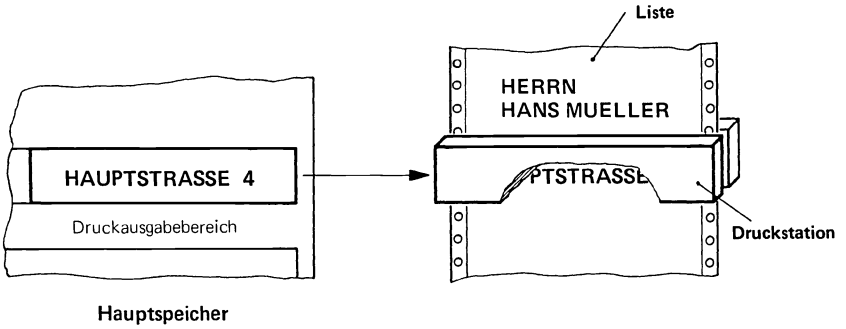


Abb. 36

Unter Berücksichtigung jener Tatsache, daß ein Computer Daten einlesen bzw. ausgeben kann, sind prinzipiell zwei Gruppen von Dateien zu unterscheiden: INPUT-Dateien und OUTPUT-Dateien (Input = Eingabe, Output = Ausgabe).
Zweifelsohne gehören die Lochkarten im Magazin des Karten-Lesers zu einer INPUT-Datei, während die LIST-Daten eine OUTPUT-Datei sind.
Damit wären aber die einführenden Hinweise zu den Dateibeschreibungen fast abgeschlossen. Ein Blick auf Abb. 37 zeigt, welche Angaben darin zu berücksichtigen sind. Als Beispiel wurde dazu zweckmäßigerweise das vertraute LIST-Programm (Abb. 30) herangezogen.
Jede Datei erhält ihre eigene Beschreibung: Als allererstes wird die benutzte Ein- oder Ausgabe-Einheit des Computers genannt, dann der Dateiname, die Art der Datei (INPUT oder OUTPUT) und die symbolische Bezeichnung des entsprechenden Einlese(oder Ausgabe)-Bereiches. Für die INPUT-Datei KART ist zusätzlich noch ein EOF *)-Merkmal mit anzugeben, dessen Sinn und Zweck am Ende dieses Kapitels erläutert wird.

Dateibeschreibung für die Lochkarten auf der Karten-Abfühleinheit

Name der Datei	KART
Name des Karteneinlesebereiches	KARBER
Art der Datei	INPUT
EOF-Merkmal	ENDE

Dateibeschreibung für die auf dem Drucker auszugebende Liste

Name der Datei	DRUCK
Name des Druckausgabebereiches	AUSGAB
Art der Datei	OUTPUT

Abb. 37

* EOF = END OF FILE = Ende des Datenbestandes

Soll nun im Programm zum Beispiel die Instruktion

GET KARBER

ausgeführt werden, so weiß der Computer aus der Dateibeschreibung, daß als KART-Datei die Lochkarten im Magazin des Karten-Lesers anzusehen sind. Auf diese Art und Weise erfährt der Dateiname die ihm zugewiesene Bedeutung. Doch damit sind die Angaben für diesen Datenbestand noch keinesfalls erschöpft. So muß unter anderem auch der Karteneinlesebereich bezeichnet werden – der Programmierer vergab dafür den symbolischen Namen KARBER. (Die Festlegung der Größe von KARBER erfolgt unter der Rubrik Definitionen). Außerdem verlangt eine formelle Vorschrift, jede Datei nach INPUT oder OUTPUT zu klassifizieren.
Eine sehr elegante Vereinfachung der Programmierung ergibt sich aus der Festlegung eines EOF-Merkmales. Dadurch kann letzlich die explizite Abfrage (Abb. 27)

Sind noch Lochkarten im Karten-Magazin?

entfallen:
(Nach jedem Lesebefehl gelangt die betroffene Lochkarte in das Ablagefach des Karten-Lesers. Dieser Vorgang wiederholt sich solange, bis das Magazin des Karten-Lesers leer ist. Kehrte jetzt das Programm erneut zu dem Lesebefehl zurück, würde Unmögliches verlangt werden, denn eine nicht vorhandene Lochkarte (das Magazin ist bereits leer) kann nicht eingelesen werden. Der Computer wäre „verwirrt". Um eine solche Situation von vornherein auszuschließen, wurde bisher (Abb. 27) vor dem Lesebefehl immer erst danach gefragt, ob sich noch Lochkarten im Karten-Magazin befinden. Eine verneinende Antwort deutete an, daß das Auflisten aller Lochkarten als beendet anzusehen ist – das Programm führte in einem solchen Fall zu der Instruktion

Arbeitsende).

Als EOF-Merkmal ist nichts weiter als der Name einer Instruktion, zum Beispiel ENDE, anzugeben. Zu dieser Instruktion verzweigt das Programm selbsttätig, wenn der Lesebefehl

GET KART

erteilt wird, aber nicht mehr ausgeführt werden kann, wenn sich also keine Lochkarte mehr im Magazin des Karten-Lesers befindet.
So vereinfacht sich das Blockdiagramm (Abb. 27) des LIST-Programmes zu der in Abb. 38 angegebenen Form: Die Abfrage

Sind noch Lochkarten im Kartenmagazin?

entfällt somit; an ihre Stelle tritt in der Beschreibung der KART-Datei das EOF-Merkmal ENDE. Da die Instruktion ENDE über den Lesebefehl

GET KART

angesteuert wird, wurde im Blockdiagramm (Abb. 38) eine gestrichelte Verbindungslinie

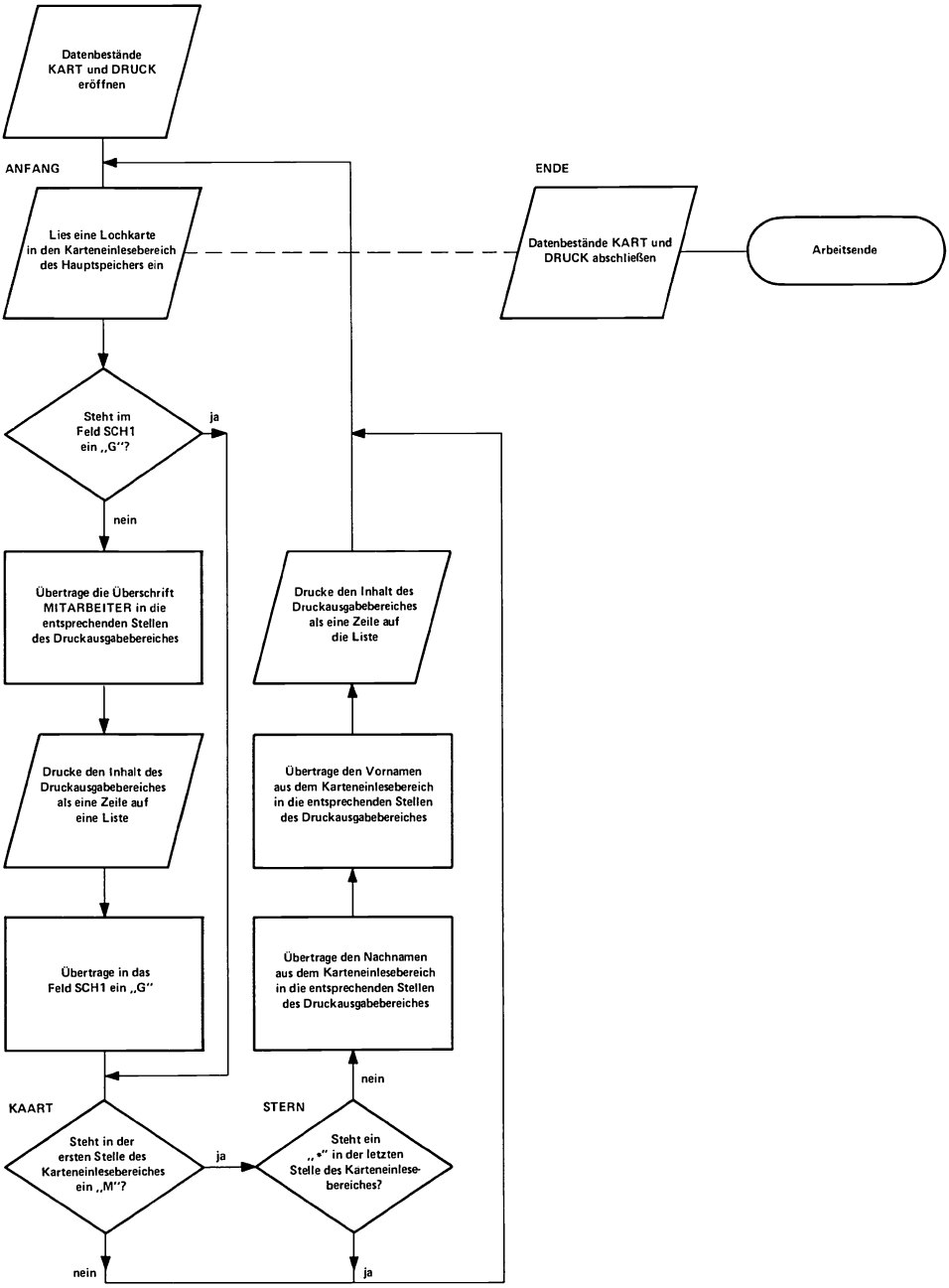

Datenbestände KART und DRUCK eröffnen
ANFANG
ENDE
Lies eine Lochkarte in den Karteneinlesebereich des Hauptspeichers ein
Datenbestände KART und DRUCK abschließen
Arbeitsende
Steht im Feld SCH1 ein „G"?
ja
nein
Übertrage die Überschrift MITARBEITER in die entsprechenden Stellen des Druckausgabebereiches
Drucke den Inhalt des Druckausgabebereiches als eine Zeile auf die Liste
Drucke den Inhalt des Druckausgabebereiches als eine Zeile auf eine Liste
Übertrage den Vornamen aus dem Karteneinlesebereich in die entsprechenden Stellen des Druckausgabebereiches
Übertrage in das Feld SCH1 ein „G"
Übertrage den Nachnamen aus dem Karteneinlesebereich in die entsprechenden Stellen des Druckausgabebereiches
nein
KAART
STERN
Steht in der ersten Stelle des Karteneinlesebereiches ein „M"?
ja
Steht ein „*" in der letzten Stelle des Karteneinlese-bereiches?
nein
ja

Abb. 38

gewählt. Diesem Weg folgt das Programm, sobald eine nicht mehr vorhandene Lochkarte eingelesen werden soll.
Im Zusammenhang mit den Dateibeschreibungen ist auch noch folgendes zu beachten: Eine Vorschrift verlangt, daß zu Beginn des Programmes alle beteiligten Dateien eröffnet werden. Dieser Forderung kommt im Blockdiagramm (Abb. 38) die Instruktion

```
OPEN    KART,DRUCK
```

nach, bei der hinter der Operationsbezeichnung OPEN die Dateinamen aufzuführen sind. Auf diese Weise wird dem Computer mitgeteilt, auf welche Datenbestände sich das Programm beziehen soll.
Wird zu einem späteren Zeitpunkt der Befehl

```
GET     KART
```

oder

```
PUT     DRUCK
```

erteilt, so kann der Computer die den Dateinamen KART oder DRUCK zugemessene Bedeutung der entsprechenden Dateibeschreibung entnehmen.
Nach dem eigentlichen Ablauf des Programmes – so fordert es die Vorschrift – müssen sämtliche Dateien „abgeschlossen" (verabschiedet) werden. Diesem Zweck dient die Instruktion

```
CLOSE   KART,DRUCK.
```

Daran anschließend ist das endgültige Ende der Arbeit mit

```
END OF JOB (Arbeitsende)
```

zu konstatieren.
Nach allen diesen Vorbetrachtungen genügt ein kleiner Schritt, um endlich zu einem richtigen symbolischen Computerprogramm zu gelangen.

8. Das symbolische Programm

Jetzt ist es endlich soweit: Das LIST-Programm (Abb. 38) soll eine computerwürdige Form erhalten. Bisher wurden Befehle wie

Steht im Feld SCH1 ein „G"?

oder

Übertrage den Vornamen aus dem Karteneinlesebereich in die entsprechenden Stellen des Druckausgabebereiches

in unserer deutschen Sprache abgefaßt. Ein Ausländer, egal ob Japaner, Eskimo oder Araber, hätte sich natürlich auch seiner eigenen Muttersprache bedient.
Es besteht kein Zweifel, daß der Computer solche Befehle im Prinzip entgegennehmen und auch ausführen könnte, doch würde er sich schwerlich auf alle Fremdsprachen dieser Welt einstellen können.
Aus diesem Grunde ist man darauf angewiesen, dem Computersystem einen Auftrag in einer besonderen Formulierung anzubieten, in Form einer weltweiten Programmiersprache (ASSEMBLER, COBOL, FORTRAN, ALGOL, PL/1 usw.). Dabei sind strenge formelle Richtlinien zu beachten. Auf diese Weise wird dem „Elektronengehirn" nichts Unmögliches abverlangt.
Die folgenden Beispiele sind auf eine der bekanntesten Zwei-Adress-Programmiersprachen, auf die ASSEMBLER-Sprache abgestellt. Die Behandlung einer solchen „maschinennahen" Sprache läßt die programmtechnische Arbeitsweise einer elektronischen Datenverarbeitungsanlage am besten erkennen.
So fremd ASSEMBLER-Instruktionen zunächst noch anmuten mögen, nach mehrmaligem Gebrauch werden sie zu einem vertrauten „Handwerkszeug". Im allgemeinen umfaßt eine Instruktion zwei Operanden, die durch ein Komma getrennt sind. Als Operanden können symbolische Namen oder auch andere Ausdrücke auftreten:

```
MVC    TIK,TAK
SP     VIEL,BUM
```

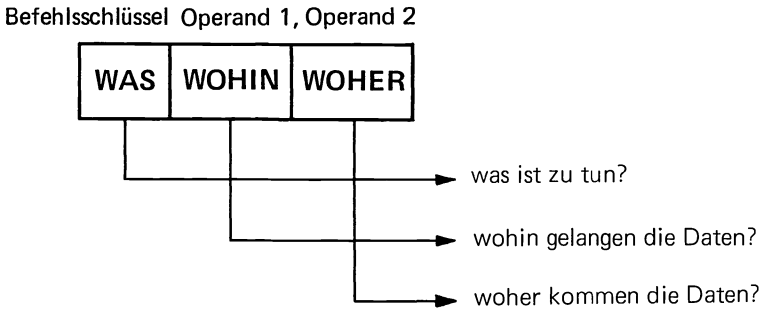
Jeder Operand bezeichnet einen bestimmten Platz (eine bestimmte Adresse) im Hauptspeicher. Durch den Befehlsschlüssel (MVC = Übertrage; SP = Subtrahiere) werden jeweils die beiden Operanden miteinander verknüpft. So bedeutet zum Beispiel

```
MVC    TIK,TAK                    (Abb. 21),
```

daß der Inhalt des Feldes TAK in das Feld TIK zu übertragen ist, während

```
SP     VIEL,BUM
```

die Subtraktion des Inhaltes des Feldes BUM von dem des Feldes VIEL bewirkt.
Eine genaue Betrachtung vieler ASSEMBLER-Instruktionen weist auf deren gleichartigen Aufbau hin (Abb. 39). Durch den Befehlsschlüssel wird die Art der Operation festgelegt (Übertragung, Vergleich, Addition usw.), während die Operanden 1 und 2 die Haupt-



Allgemeiner Aufbau einer Instruktion

Abb. 39

speicherfelder bezeichnen, auf deren Inhalt sich die Operation beziehen soll.
Bei einer Übertragung gelangt der Inhalt des zweiten Operandenfeldes in das erste. Bei einer Subtraktion hingegen wird der Inhalt des zweiten Operandenfeldes vom Inhalt des ersten abgezogen, wobei das Ergebnis im ersten Operandenfeld erscheint.
Im Grunde genommen stellen diese ASSEMBLER-Instruktionen kein Neuland dar, wenn man einmal von deren formeller Schreibweise absieht. Es wäre allerdings sträflich, wollte sich ein Programmierer großzügig über diesen Formalismus hinwegsetzen. Ein fehlendes Komma oder ein überflüssiger Punkt in einer Instruktion sind oft die Ursachen für ein fehlerhaftes Programm.
Nun soll das LIST-Programm in der ASSEMBLER-Sprache formuliert werden. Zuvor sei allerdings wegen der besseren Übersicht das Zeilen-Schema (Abb. 40a) nach dem Blockdiagramm (Abb. 38). erstellt. Darin finden diesmal auch die Dateibeschreibungen Aufnahme.
Direkt neben dem Zeilen-Schema (Abb. 40a) mit dem beschreibenden Text ist das Zeilen-Schema mit den entsprechenden ASSEMBLER-Instruktionen (Abb. 40b) aufgeführt. So ist leicht zu erkennen, welche Bedeutung die einzelnen ASSEMBLER-Instruktionen haben (eine detaillierte Erläuterung dazu ist in der Aufstellung nach Abb. 41 angegeben; darin sind der Vollständigkeit halber auch Instruktionen enthalten, die im LIST-Programm (Abb. 40) nicht benutzt werden. Da bloße Worte zumeist etwas trocken klingen, ist die Wirkungsweise der Instruktionen in Abb. 42 auch noch auf graphische Art und Weise illustriert).
Ein erfahrener Programmierer wird mitunter bereits im Blockdiagramm ASSEMBLER-Instruktionen verwenden. Diese besitzen für ihn bereits eine so große Aussagekraft, wie für einen unerfahrenen Laien die Erklärungen in der Umgangssprache (Abb. 40a). Dadurch bleibt dem Programmierer viel Schreibarbeit erspart; aufgrund einer gewissen Routine kommt er schneller zum Ziel.
Als nächstes ist das Zeilen-Schema in Abb. 40b nach einer strengen Vorschrift in sogenannte Codierblätter (Abb. 43) einzutragen. Diese Codierblätter sehen für den Namen, die Operation (Befehlsschlüssel) und die Operanden besondere Felder vor. Selbstverständlich müssen auch die Angaben der Dateibeschreibungen mit in das Codierblatt aufgenommen werden. Wie dies aber im einzelnen geschieht, ist hier für das Verständnis unwesentlich. Aus diesem Grunde sind die Dateibeschreibungen in Abb. 43 ohne formellen Ritus aufgeführt.

Dateibeschreibungen
Für die Lochkarten auf der Kartenabfühleinheit

Name der Datei	KART
Name des Karten-einlesebereiches	KARBER
Art der Datei	INPUT
EOF-Merkmal	ENDE

Für die auf dem Drucker auszugebende Liste

Name der Datei	DRUCK
Name des Druck-ausgabebereiches	AUSGAB
Art der Datei	OUTPUT

Name	Instruktion	Name	Operation	Operanden
	Datenbestände KART und DRUCK eröffnen		OPEN	KART, DRUCK
ANFANG	Lies eine Lochkarte in den Karteneinlese-bereich des Hauptspei-chers ein	ANFANG	GET	KART
	Steht im Feld SCH1 ein „G"?		CLI	SCH1,C'G'
	Verzweige nach KAART bei ja		BE	KAART
	Übertrage die Über-schrift MITARBEITER in die entsprechenden Stellen des Druckaus-gabebereiches		MVC	AUSGAB+9(11),UEBER
	Drucke den Inhalt des Druckausgabebereiches als eine Zeile auf die Liste		PUT	DRUCK
	Übertrage in das Feld SCH1 ein „G"		MVI	SCH1,C'G'
KAART	Steht in der ersten Stelle des Karteneinlesebereiches ein „M"?	KAART	CLI	KARBER,C'G'
	Verzweige nach Stern bei ja		BE	STERN
	Verzweige nach ANFANG		B	ANFANG
STERN	Steht ein „*" in der letzten Stelle des Karteneinlesebereiches?	STERN	CLI	KARBER+79,C'*'

Name	Instruktion	Name	Operation	Operanden
	Verzweige nach ANFANG bei ja		BE	ANFANG
	Übertrage den Nachnamen aus dem Karteneinlesebereich in die entsprechenden Stellen des Druckausgabebereiches		MVC	AUSGAB+3(16),KARBER+1
	Übertrage den Vornamen aus dem Karteneinlesebereich in die entsprechenden Stellen des Druckausgabebereiches		MVC	AUSGAB+19(11),KARBER+17
	Drucke den Inhalt des Druckausgabebereiches als eine Zeile auf die Liste		PUT	DRUCK
	Verzweige nach ANFANG		B	ANFANG
ENDE	Datenbestände KART und DRUCK abschließen.	ENDE	CLOSE	KART, DRUCK
	Arbeitsende		EOJ	
	Definitonen:		Definitionen:	
KARBER	80 Bytes, Inhalt: Blank	KARBER	DC	CL80' '
AUSGAB	30 Bytes, Inhalt: Blank	AUSGAB	DC	CL30' '
SCH1	1 Byte , Inhalt: 'A'	SCH1	DC	CL1'A'
UEBER	11 Bytes, Inhalt: 'MITARBEITER'	UEBER	DC	C'MITARBEITER'

Abb. 40a

ASSEMBLER-Sprache

Abb. 40b

Instruktion	Wirkungsweise der Instruktion	Bemerkungen
CLI SCH1,C'G'	Der Inhalt der mit SCH 1 bezeichneten Hauptspeicherstelle wird mit dem 1 Byte umfassenden Direktoperanden G verglichen.	Der Direktoperand muß immer im 2. Operanden hinter einem C in Hochkommas eingeschlossen sein. Das C weist den Direktoperanden als **C**haracter (= Zeichen) aus.
MVI SCH1,C'G'	In die mit SCH1 bezeichnete Hauptspeicherstelle wird der 1 Byte umfassende Direktoperand G übertragen.	Der Direktoperand muß immer im 2. Operanden hinter einem C in Hochkommas eingeschlossen sein. Das C weist den Direktoperanden als **C**haracter (= Zeichen) aus.
MVC AUSGAB+3(16),KARBER+1	Der Inhalt des mit KARBER bezeichneten Feldes wird ab der Stelle KARBER+1 in einer Länge von 16 Bytes nach AUSGAB+3 und die daran anschließenden Stellen übertragen.	Die Anzahl der zu übertragenden Zeichen ist in Klammern gesetzt im 1. Operanden anzugeben. Der 1. und 2. Operand umfassen zumeist symbolische Feldnamen. Falls erforderlich, dürfen diese Namen zur relativen Adreßrechnung herangezogen werden (Abb. 34). Bei fehlender Längenangabe umfaßt die Übertragung immer so viele Bytes, wie sie laut Definition dem als 1. Operanden genannten Feld zugestanden wurden.
MVC RUF,=C'SOS'	In das mit RUF bezeichnete Hauptspeicherfeld (es soll laut Definition eine Größe von 3 Bytes umfassen) wird der 3 Byte umfassende Direktoperand SOS übertragen.	Der Direktoperand muß immer im 2. Operanden hinter einem =C in Hochkommas eingeschlossen sein. Er kann mehrere Zeichen umfassen. Das C weist den Direktoperanden als **C**haracter aus. Die Gleichheitszeichen = sind formell vorgeschrieben.

Instruktion	Wirkungsweise der Instruktion	Bemerkungen
CLC CE+1(2),AU+3	Der Inhalt des mit CE bezeichneten Feldes wird ab der Stelle CE+1 in einer Länge von 2 Bytes mit dem Inhalt der mit AU+3 bezeichneten und der daran anschließenden Stelle des Feldes AU verglichen. (Die Felder CE und AU sollen laut Definition eine Größe von je 6 Bytes umfassen).	Die Anzahl der zu vergleichenden Zeichen ist in Klammern im 1. Operanden anzugeben. Der 1. und 2. Operand umfassen zumeist symbolische Feldnamen. Falls erforderlich, dürfen diese Namen zur relativen Adreßrechnung herangezogen werden. Bei fehlender Längenangabe bezieht sich der Vergleich auf eine solche Feldlänge, wie sie das als 1. Operand genannte Feld laut Definition aufweist.
CLC WE+1(2),=C'12'	Der Inhalt des mit WE bezeichneten Feldes wird ab der Stelle WE+1 in einer Länge von 2 Bytes mit dem 2 Bytes umfassenden Direktoperanden 12 verglichen.	Der Direktoperand muß immer im 2. Operanden hinter einem =C in Hochkommas eingeschlossen sein. Er kann mehrere Zeichen umfassen. Das C weist den Direktoperanden als Charakter (= Zeichen) aus. Die Gleichheitszeichen = sind formell vorgeschrieben
B ANFANG	Das Programm verzweigt zu der Instruktion mit dem Namen ANFANG.	Bei einer unbedingten Verzweigung ist stets der Name der Instruktion anzugeben, zu der das Programm verzweigen soll.
BE STERN	Das Programm verzweigt zu der Instruktion mit dem Namen STERN, wenn ein vorausgehender Vergleich mit der Vergleichsentscheidung „gleich" endet.	Bei einer bedingten Verzweigung, die sich immer an einen Vergleich anschließt, ist stets der Name der Instruktion anzugeben, bei der das Programm im Falle der Vergleichsentscheidung „gleich" fortgesetzt werden soll.

Instruktion			Wirkungsweise der Instruktion	Bemerkungen
	EOJ		Der Computer beendet die Ausführung eines Programmes.	Das Aufleuchten eines Lämpchens „WARTE-STATUS" deutet an, daß der Computer ein Programm ausgeführt hat (END OF JOB) und zur Ausführung eines anderen bereit ist.
KARBER	DC	CL80' '	Mit dieser Anweisung wird ein 80 Bytes umfassendes Feld namens KARBER mit dem Inhalt Blank definiert und im Hauptspeicher reserviert.	Als feststehende Ausdrücke sind in einer solchen Definitionsinstruktion ... DC CL.... '..' vorgeschrieben. An den mit Punkten gekennzeichneten Stellen hat der Programmierer den gewählten symbolischen Feldnamen, die Feldlänge (= Anzahl der Bytes) und das Zeichen anzugeben, welches in jedem Byte gespeichert werden soll. Als Zeichen ist in diesem Falle eine Leerstelle (Blank mit der Bitkombination 0100 0000) vorgesehen.
TIM	DS	CL5	Mit dieser Anweisung wird ein 5 Bytes umfassendes Feld namens TIM definiert und im Hauptspeicher reserviert.	Als feststehende Ausdrücke sind in einer solchen Definitionsinstruktion ... DC CL... vorgeschrieben. An den mit Punkten gekennzeichneten Stellen hat der Programmierer den gewählten symbolischen Feldnamen und die Feldlänge (= Anzahl der Bytes) anzugeben. Der Inhalt dieses Feldes ist beim Laden des Programmes unbestimmt und kann beliebige Zeichen umfassen, wie sie aufgrund des zuvor abgelaufenen Programmes an den entsprechenden Stellen gespeichert wurden.

Instruktion	Wirkungsweise der Instruktion	Bemerkungen
UEBER DC C'MITARBEITER'	Mit dieser Anweisung wird ein Feld namens UEBER mit dem Inhalt MITARBEITER definiert und im Hauptspeicher reserviert.	Als feststehende Ausdrücke sind in einer solchen Definitionsinstruktion . . . DC C'. . . . .' vorgeschrieben. An den mit Punkten gekennzeichneten Stellen hat der Programmierer den gewählten symbolischen Feldnamen und alle Zeichen anzugeben, die in diesem Feld gespeichert werden sollen (die Feldlänge ermittelt der Computer von selbst aus der Anzahl der Zeichen).

Mnemonischer Operationsschlüssel	Bedeutung
CLC	Compare Logic Character = Vergleiche zeichenweise
CLI	Compare Logic Immediately = Vergleiche mit einem direkt angegebenen Zeichen
MVC	Move Character = Übertrage Zeichen
MVI	Move Immediately = Übertrage ein direkt angegebenes Zeichen
BE	Branch on Equal = Verzweige bei gleich
B	Branch = Verzweige
DC	Define Character = Definiere Zeichen
DS	Define Storage = Definiere Speicherplatz

Abb. 41

Inhalt der Hauptspeicherstellen vor Ausführung der Instruktion	Instruktion	Inhalt der Hauptspeicherstellen nach Ausführung der Instruktion
A SCH1	CLI SCH1,C'G'	A SCH1
A SCH1	MVI SCH1,C'G'	G SCH1
KOS AUSGAB EMUELL KARBER	MVC AUSGAB+3(16),KARBER+1	MUE AUSGAB EMUELL KARBER
RUF	MVC RUF,=C'SOS'	SOS RUF
HOKUS- CE POKUS- AU	CLC CE+1(2),AU+3	HOKUS- CE POKUS- AU
0123AB WE	CLC WE+1(2),=C'12'	0123AB WE
ABCDEFGH YZ12345678	KARBER DC CL80''	KARBER
XYZABC	TIM DS CL5	XYZABC TIM
LNOPQRSTUVWX	UEBER DC C'MITARBEITER'	MITARBEITERX UEBER

Die Hauptspeicherfelder sind entsprechend ihrer definierten Größe stark umrandet.

Abb. 42

Nach der Eintragung aller Befehle und Definitonsanweisungen in das Codierblatt liegt das LIST-Programm nunmehr in seiner codierten Form (Abb. 43) vor.
Doch damit weiß der Computer immer noch nichts anzufangen; für ihn wären entsprechende (Programm-)Lochkarten viel interessanter.
Deshalb wird der Inhalt einer jeden Zeile des Codierblattes spaltengerecht in eine Lochkarte übertragen (Abb. 44). So etwas war ohnehin zu erwarten, denn eine Zeile des

Codierblattes ist nicht ohne Grund in genau 80 Positionen (gemäß den 80 Spalten einer Lochkarte) aufgeteilt. Sobald alle Programm-Lochkarten vorliegen, kann man von einem symbolischen Programm (Abb. 44) sprechen. Wie der Computer damit umgeht, wird im folgenden Kapitel 9 näher beschrieben.
Doch rufen wir uns nochmals die verschiedenen Schritte ins Gedächtnis zurück, die zu einem symbolischen Programm führen:

1. Aufstellung des Blockdiagrammes entsprechend der Aufgabenstellung.

2. Übertragung der Programmbefehle etc. in Codierblätter.

3. Ablochen der Codierblätter zum symbolischen Programm.

IBM Assembler

Programm	MITARBEITER - LISTE	Lochhinweise		Problem	MIL
		Zeichen			
Programmierer	Datum	Lochung		Seite	01 von 2

Name	Operation	Operand und Bemerkungen	Karte Nr.
		Dateibeschreibungen	010
		Für die Lochkarten auf der Kartenabfühleinheit	020
		Name der Datei KART	030
		Name des Karten-einlesebereiches KARBER	040
		Art der Datei INPUT	050
		EOF-Merkmal ENDE	060
		Für die auf dem Drucker auszugebende Liste	070
		Name der Datei DRUCK	080
		Name des Druck-ausgabebereiches AUSGAB	090
		Art der Datei OUTPUT	100
	OPEN	KART,DRUCK	110
ANFANG	GET	KART	120
	CLI	SCH1,C'G'	130
	BE	KAART	140
	MVC	AUSGAB+9(11),UEBER	150
	PUT	DRUCK	160
	MVI	SCH1,C'G'	170
KAART	CLI	KARBER,C'E'	180
	BE	STERN	190
	B	ANFANG	200
STERN	CLI	KARBER+79,C'*'	201
	BE	ANFANG	202

Abb. 43a

IBM Assembler

Programm		Lochhinweise		Problem M I L (73–75)
		Zeichen		Seite 02 (76–77) von 8
Programmierer	Datum	Lochung		

Name (8)	Operation (10–14)	Operand und Bemerkungen (20–70)	Karte Nr. (78–80)
	MVC	AUSGAB+3(16),KARBER+1	010
	MVC	AUSGAB+19(11),KARBER+17	020
	PUT	DRUCK	030
	B	ANFANG	040
ENDE	CLOSE	KART,DRUCK	050
	EOJ		060
**DEFINITIONEN			070
KARBER	DC	CL80' '	080
AUSGAB	DC	CL30' '	090
SCH1	DC	CL1'A'	100
UEBER	DC	C'MITARBEITER'	110
			120
			130
			140
			150
			160
			170
			180
			190
			200

Abb. 43b

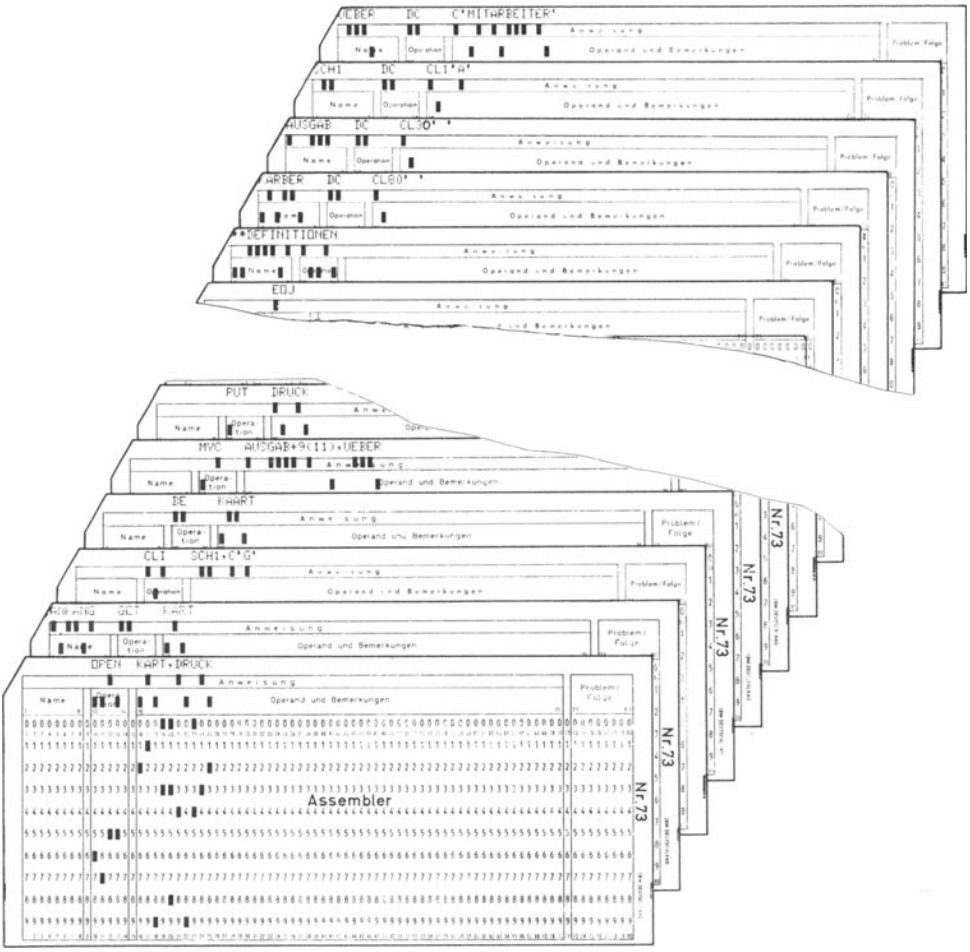

UEBER DC C'MITARBEITER'
SCHI DC CL1'A'
AUSGAB DC CL30' '
ARBER DC CL80' '
*DEFINITIONEN
EOJ
PUT DRUCK
MVC AUSGAB+9(11),UEBER
DE KARTE
CLI SCHI,C'G'
OPEN KARTE,DRUCK
Name
Operation
Anweisung
Operand und Bemerkungen
Problem/Folge
Nr.73
Assembler

Abb. 44

9. Die Umwandlung eines symbolischen in ein echtes Programm

Nachdem jetzt das symbolische LIST-Programm vorliegt, darf man sich glücklich schätzen, denn alles, was noch unternommen werden muß, um den Computer zur Ausgabe der gewünschten Mitarbeiter-Liste zu bewegen, ist reine Routine.
Im Kapitel 6 wurde bereits kurz darauf hingewiesen, daß symbolische Bezeichnungen zwar für den Programmierer eine recht anschauliche Aussagekraft besitzen, jedoch für den Computer so ziemlich nichtssagend sind. Denn der Computer kann nun einmal nur mit Adressen (und anderen verschlüsselten Angaben) umgehen, so wie diese bei einem echten Maschinenprogramm in der Maschinensprache auch vorliegen.
Das sollte jedoch den Programmierer nicht etwa veranlassen, auf die Benutzung des ASSEMBLERS von vornherein zu verzichten und selbst in der Maschinensprache zu programmieren. Ein solches Vorhaben wäre für ihn mitunter viel zu schwierig und zeitraubend. Doch wie kann dann die Kluft zwischen ASSEMBLER und Maschinensprache überbrückt werden?
Die Antwort läßt sich bereits erahnen: Man schaltet einen automatischen Übersetzer ein und überträgt ihm die gewünschte Umwandlung des symbolischen LIST-Programmes. Dieser Übersetzer heißt in Fachkreisen ASSEMBLER-Compiler. Er ist für alle symbolischen ASSEMBLER-Programme in gleicher Weise universell verwendbar.
Der Compiler ist nicht etwa ein leibhaftiger Dolmetscher, sondern ein Programm. Dieses wird von der Herstellerfirma der Datenverarbeitungsanlage, zum Beispiel in Form eines Lochkartenpaketes geliefert. Mit Hilfe des Compilers ist es dann möglich, innerhalb weniger Minuten aus dem symbolischen LIST-Programm ein echtes, verschlüsseltes Maschinenprogramm zu erzeugen. Was dabei zu beachten ist, geht aus dem folgenden Rezept eines Umwandlungslaufes hervor. Zuvor sei allerdings noch bemerkt, daß das Computersystem für eine solche Umwandlung als Mindestausrüstung neben der Zentraleinheit einen Karten-Leser, einen Karten-Stanzer und einen Drucker umfassen muß (Abb. 45).

Hinweise für einen Umwandlungslauf

1. In den Drucker eine Liste einspannen.

2. In das Magazin des Karten-Stanzers leere Lochkarten einlegen.

3. Das Lochkartenpaket des Compiler-Programmes in das Magazin des Karten-Lesers legen. Taste „LADEN" drücken (dadurch wird das Laden des Umwandlungsprogrammes in den Hauptspeichers ausgelöst, die gelesenen Compiler-Lochkarten gelangen nacheinander in das Ablagefach des Karten-Lesers).

4. Das symbolische (LIST)-Programm in das Magazin des Karten-Lesers legen. Taste „START" drücken. Die symbolischen Programm-Lochkarten werden nacheinander gelesen. Durch die Wirkung des Compilers werden deren symbolische Angaben in die echte Maschinensprache übersetzt und die echten Maschinenbefehle in (Programm-)-Lochkarten gestanzt.

5. Das echte (LIST)-Programm kann dann dem Ablagefach des Karten-Stanzers entnommen werden.

Während des Umwandlungslaufes erfolgt über den Drucker die Ausgabe einer sogenannten Umwandlungsliste. Darauf werden unter anderem neben den symbolischen Instruktionen auch die echten Maschinenbefehle vermerkt. Diese Hinweise sind für eine spätere Feststellung eventueller Programmfehler unerläßlich.
Über den Karten-Stanzer (Abb. 45) ist bisher noch nichts erwähnt worden. Dessen Funktionsweise ist der des Karten-Lesers „entgegengesetzt". Durch einen Ausgabebefehl (z. B. PUT STANZ) werden auf „einen Schlag" Angaben in eine zuvor noch leere Lochkarte gestanzt. Diese Angaben entstammen dem Stanz-Ausgabebereich des Hauptspeichers. Die zunächst noch leere Lochkarte gelangt aus dem Magazin unter die Stanzstation und nachdem sie dort mittels vieler kleiner Stempelchen mit den entsprechenden Lochungen versehen wurde, fällt sie in das Ablagefach.
Über diesen Karten-Stanzer wird während des Umwandlungslaufes das Programm in der

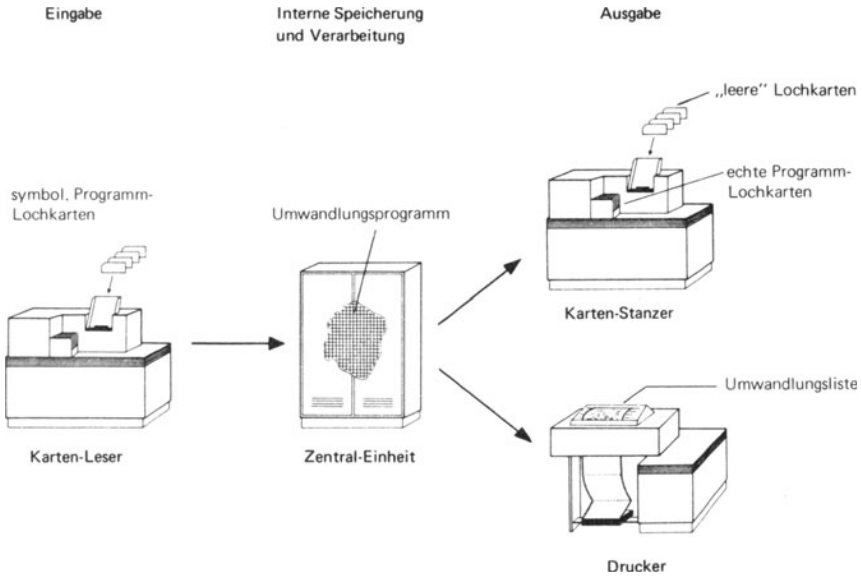


Abb. 45

Maschinensprache (Objekt-Programm) ausgegeben. Eine detaillierte Kenntnis des Compilers ist nicht erforderlich; dem Benutzer genügt es zu wissen, daß der Compiler das successive Einlesen des symbolischen Kartenpaketes (Quellen-Programm), die Umwandlung der symbolischen Instruktionen in die Maschinensprache und die Ausgabe der gestanzten Objekt-Programm-Lochkarten besorgt.
Bei einer genaueren Überlegung fragt man sich allerdings, wie wohl der Compiler mit den Angaben in der Dateibeschreibung verfährt.
Die bloße Nennung eines Dateinamens in einem Lese- oder Schreibbefehl, wie GET KART oder PUT DRUCK wäre ziemlich nichtssagend, würde dieser Name nicht in der entsprechenden Dateibeschreibung eine Erklärung finden. Aus diesem Grunde werden auch von dem Compiler bei der Umwandlung einer Ein-/Ausgabeinstruktion (Lese-/Schreibinstruktion) alle wichtigen Dateiangaben berücksichtigt.

Die Anweisung GET KART zum Beispiel würde demnach in der echten Maschinensprache Hinweise enthalten, die

a) den Karten-Leser zum Einlesen einer Lochkarte veranlassen, wobei
b) der Inhalt der Lochkarte in den 80 Byte großen Karteneinlesebereich KARBER übertragen wird.

Verständlich, daß zur Erfüllung einer solchen Forderung mehrere maschinen-interne Schritte erforderlich sind und erfreulich, daß der Compiler diese Schritte von selbst angibt.
Nach erfolgter Umwandlung kann das Objekt-Deck (Deck = Lochkartenstapel) des LIST-Programmes dem Ablagefach des Karten-Stanzers entnommen werden. Diese echten Programm-Lochkarten sind schonend zu behandeln und stets griffbereit aufzubewahren; denn nach ihnen soll der Computer vielleicht viele tausend Male Mitarbeiter-Listen erstellen.
Sobald für eine beliebige Menge von MITARBEITER-LOCHKARTEN (Abb. 24) eine solche Liste gewünscht wird, ist das Objekt-Deck des LIST-Programmes über den Karten-Leser in den Hauptspeicher zu laden. Anschließend können die Mitarbeiter-Lochkarten gelesen und gelistet werden; so wie es das Programm vorschreibt.
Ohne die echte Maschinensprache wären die komplizierten elektronischen Schaltungen des Computers niemals zu einer Ausführung der einzelnen Befehle zu bewegen. Die Umwandlung eines symbolischen Programmes ist schon aus diesem einfachen Grunde unumgänglich.
Wie bereits erwähnt, sind für die Befolgung eines Ein-/Ausgabe-Befehles mehrere maschineninterne Schritte erforderlich. Diese Schritte werden von einer Ein-/Ausgabe-Steuerung (I/O-Control; I wie Input = Eingabe, O wie Output = Ausgabe) ausgeführt. Sie garantiert eine einwandfreie Datenein- bzw. -ausgabe und leistet somit eine wertvolle Hilfe bei der Ausführung eines Programmes.

10. Zahlensysteme

Es klingt wohl zunächst etwas verwunderlich, wenn man erfährt, daß dem Aufbau einer mehrstelligen Zahl ein bestimmtes System zugrunde liegen soll. Was unser bewährtes Dezimalsystem anbelangt, so erfordert dessen Gebrauch natürlich keinerlei Gedanken mehr; alles ist uns dank vertrauter Gewohnheit in Fleisch und Blut übergegangen: Die Benutzung der Ziffernsymbole 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 und die Aufgliederung einer Zahl in Stellen, in sogenannte Wertstufen wie Einer, Zehner, Hunderter, Tausender. In jeder dieser Stellen steht immer eines der zehn verschiedenen Ziffernsymbole (Abb. 46). Mit diesen kurzen Hinweisen ist jedoch der allgemeine Aufbau von Zahlen nach der Stellenschreibweise bereits umrissen.

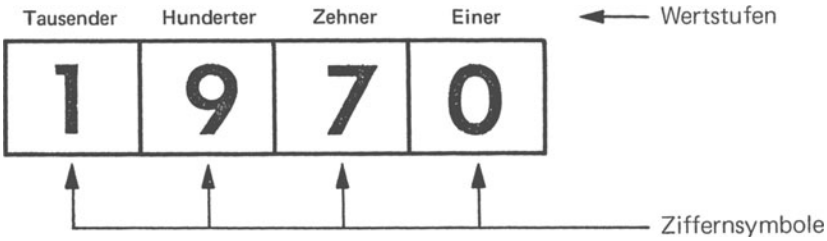


Abb. 46

Zu Zeiten der alten Ägypter war eine solche Stellenschreibweise noch völlig unbekannt – jede Zahl hatte ihr spezielles Zeichen. Die alten Ägypter wußten damit natürlich gut umzugehen, doch sicherlich hätten sie sich auch für das Dezimalsystem entschieden, wenn sie nur etwas davon geahnt hätten.
Vielleicht wäre ihre Entscheidung aber auch zugunsten eines anderen Zahlensystems mit mehr oder weniger als zehn verschiedenen Ziffernsymbolen ausgefallen, zum Beispiel für das Dualsystem. Dieses Dualsystem kennt nur die beiden Ziffernsymbole O und L. Dabei bezeichnet O die Anzahl 0 und die L die Anzahl 1. Dennoch ist es möglich, beliebig große Punktmengen mit Hilfe dieser beiden Symbole als mehrstellige Dualzahl zu erfassen: In Abb. 47 sind für eine stetig ansteigende Punktmenge einmal die Dezimal- und zum anderen die Dualzahlen angeben.
Für eine Menge von beispielsweise .....Punkten hat die Dezimalzahl 021 die gleiche Aussagekraft wie die Dualzahl LOLOL. Unbestritten liegt in der 021 wohl die größere Eleganz der Darstellung, doch LOLOL bietet für das Dualsystem die einzige Möglichkeit zur zahlenmäßigen Erfassung jener Punktmenge. Die Ausdeutung dieser Dualzahl ist denkbar einfach (Abb. 48):
Die aufsteigenden Zahlen in Abb. 47 sind nicht das Ergebnis willkürlichen Vorgehens, sondern eines bestimmten Schemas. Betrachtet man die untereinanderstehenden Dezimalzahlen, so stellt man in der Spalte der Einerstelle eine bestimmte Periodizität der Ziffernsymbole 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 fest. In der Spalte der Zehnerstelle hingegen folgt auf zehnmal die „0" zehnmal die „1", zehnmal die „2" ... zehnmal die „9". In der Spalte der Hunderterstelle werden die Intervalle noch größer: Hundertmal (= 10x10) die „0", hundertmal (= 10x10) die „1", hundertmal (= 10x10) die „2", ... und in der Spalte der Tausender erscheint zuerst tausendmal (= 10x10x10) die „0", dann tausendmal (= 10x10x10) die „1", gefolgt von tausendmal (= 10x10x10) die „2" ..... usw.

Dezimalsystem	Punktmenge	Dualsystem
000		O O O O O O O O
001	·	O O O O O O O L
002	··	O O O O O O L O
003	···	O O O O O O L L
004	····	O O O O O L O O
005	·····	O O O O O L O L
006	·····	O O O O O L L O
007	·····	O O O O O L L L
008	·····	O O O O L O O O
009	·····	O O O O L O O L
010	·····	O O O O L O L O
011	·····	O O O O L O L L
012	·····	O O O O L L O O
013	·····	O O O O L L O L
014	·····	O O O O L L L O
015	·····	O O O O L L L L
016	·····	O O O L O O O O
017	·····	O O O L O O O L
018	·····	O O O L O O L O
019	·····	O O O L O O L L
020	·····	O O O L O L O O
021	·····	O O O L O L O L
022	·····	O O O L O L L O
023	·····	O O O L O L L L
024	·····	O O O L L O O O
⋮		⋮
096		O L L O O O O O
097		O L L O O O O L
098		O L L O O O L O
099		O L L O O O L L
100		O L L O O L O O
101		O L L O O L O L
102		O L L O O L L O
103		O L L O O L L L
104	(ensprechend viele Punkte)	O L L O L O O O
105		O L L O L O O L
⋮		⋮
196		L L O O O L O O
197		L L O O O L O L
198		L L O O O L L O
199		L L O O O L L L
200		L L O O L O O O
201		L L O O L O O L
⋮		⋮

Anmerkung: „Führende" Nullen (0 bzw. O) wurden nur aus Gründen der Übersichtlichkeit angegeben.

Abb. 47

Überall tritt die „10" als charakteristische Größe auf. Sie entspricht genau der Anzahl der verschiedenen Ziffernsymbole (0, 1, 2, ... 9) im Dezimalsystem. Von Wertstufe zu Werstufe vergößert sich in den Spalten das Intervall gleichartiger Ziffern um den Faktor 10.
Ein analoges Schema gilt im übertragenen Sinne auch für das Dualsystem; jedoch treten hier nicht zehn, sondern nur zwei verschiedene Ziffernsymbole (O und L) auf. Folglich vergrößert sich auch das Intervall gleicher Dual-Ziffern von Wertstufe zu Wertstufe nur um den Faktor 2. In der Spalte der ersten Wertstufe *) wechseln O und L einander ab. In der zweiten Wertstufe folgen auf zweimal die „O" zweimal das „L" usw., während in der Spalte der dritten Wertstufe viermal (= 2x2) die „O" und viermal (= 2x2) das „L" einander abwechseln. In der Spalte der vierten Wertstufe umfaßt der Zyklus schließlich achtmal (= 2x2x2) die „O" und achtmal (= 2x2x2) das „L".
Die Anwendung dieses Schemas macht es höchst einfach, aufsteigende Dualzahlen niederzuschreiben. Um die Verwendung der ungewohnten Zeichen O und L zu umgehen, hat man sich darauf geeinigt, die im Dezimalsystem üblichen Symbole 0 und 1 zu verwenden. In dieser neuen Schreibweise wird aus

LOLOL	$10101_{(2)}$	O (Null) = 0 1 (Eins) = 1

Es ist selbstverständlich, daß in einer Dualzahl nur Nullen und Einsen auftreten dürfen. Um Verwechslungen zwischen dem Dezimal- und Dualsystem von vornherein auszuschließen, wird eine Dualzahl zusätzlich durch den Index (2) gekennzeichnet. Danach hätte zum Beispiel $101_{(2)}$ die Bedeutung der Dualzahl LOL (entsprechend einer Menge von ..... Punkten). Ohne den $\text{Index}_{(2)}$ würde 101 eine Dezimalzahl für die Punktmenge hundertundeins sein.
Dank unserer Gewöhnung empfindet man nur bei Dezimalzahlen eine förmlich bildliche Größenvorstellung. Aus diesem Grunde ist auch der Wunsch verständlich, für vorliegende Dualzahlen, zum Beispiel $1011_{(2)}$ den dezimalen Wert - in diesem Falle 11- zu ermitteln. Wie man dabei vorgeht, ist in Abb. 49 dargestellt: Jede Stelle der Dualzahl ist mit einem gedachten Wertfaktor behaftet. Diese Wertfaktoren sind (für die Stellen von rechts nach links) steigende Potenzen zur Basis 2: $2^0=1$, $2^1=2$, $2^2=4$, $2^3=8$ usw. Bei der Ermittlung der dezimalen Größe einer Dualzahl ist pro Stelle die dort erscheinende Ziffer (0 oder 1) mit dem zugehörigen Wertfaktor zu multiplizieren. Die Addition aller Teilprodukte ergibt dann den gesuchten Dezimalwert.
Im Zusammenhang mit dem Computer kommt den Dualzahlen eine recht praktische Bedeutung zu. Schließlich kann man auch Zahlen im Hauptspeicher mit Hilfe ein- oder ausgeschalteter Speicherzellen darstellen. Nichts ist naheliegender, als dem Zustand solcher (nebeneinanderliegenden) Speicherzellen die Bedeutung einer Dualzahl beizumessen (Abb. 50). Eine eingeschaltete Speicherzelle entspricht einer („binären") Eins, eine ausgeschaltete Speicherzelle einer („binären") Null (siehe Kapitel 4).
Mit solchen Dualzahlen kann der Computer wie ein alter Routinier umgehen (Kapitel 12, 13).

* Die Wertstufen zählen von rechts nach links

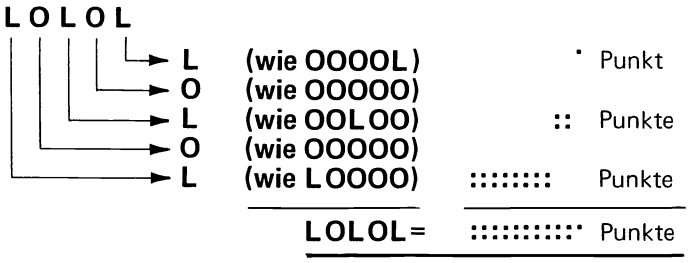


Abb. 48

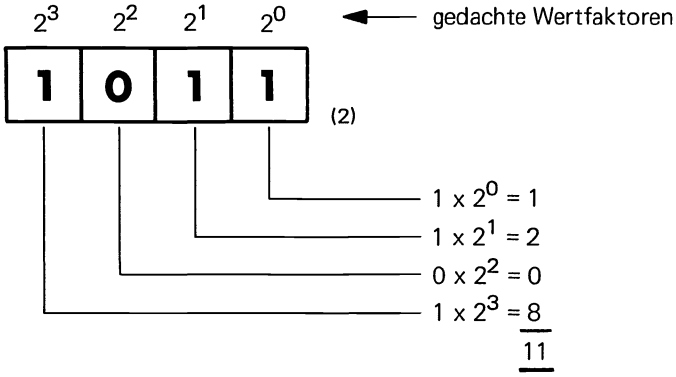
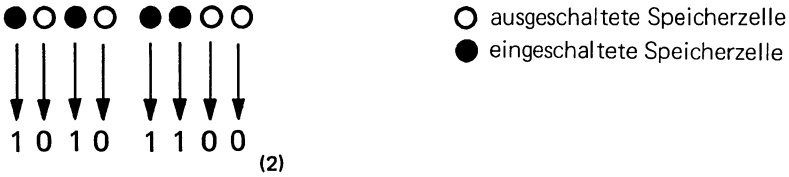


Abb. 49



Darstellung der Dualzahl $1010\ 1100_{(2)}$
mit Hilfe ein- und ausgeschalteter Speicherzellen

Abb. 50

11. Darstellungsmöglichkeiten für Hauptspeicherinformation

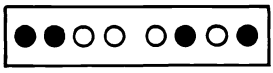
Zur Erklärung der Wirkungsweise verschiedener Instruktionen ist es oft vorteilhaft, den Inhalt von Hauptspeicherzellen auf unterschiedliche Art und Weise darzustellen.
Wie bereits bekannt, kann jedes Byte Informationen in Form eines verschlüsselten Zeichens aufnehmen. Dabei schalten elektrische Impulse die Speicherzellen (Bits) eines Bytes ein oder aus, so wie es die Verschlüsselung des Zeichens verlangt (Abb. 51). Die Information, die in einer solchen Hauptspeicherstelle steht, kann nun bit-, byte- oder halbbyteweise dargestellt werden.

○ ausgeschaltete Speicherzelle
● eingeschaltete Speicherzelle

Verschlüsselung des Zeichens „E" in einer Hauptspeicherstelle

Abb. 51

1. Die bitweise Darstellung
   Für die aus- oder eingeschalteten Speicherzellen eines Bytes werden „binäre" Nullen oder Einsen angegeben (Abb. 52).



→ 1100 0101

Abb. 52

2. Die byteweise (charakterweise) Darstellung
   Die vorliegende Bitkombination innerhalb des Bytes wird durch das entsprechende Zeichen gekennzeichnet (Abb. 53).

Abb. 53

3. Die halbbyteweise (hexadezimale) Darstellung
   Für die vier Bits eines jeden Halbbytes wird das zugehörige hexadezimale Ziffernsymbol (siehe Abb. 55) vermerkt (Abb. 54).
   Man geht dabei von der Vorstellung aus, diese vier Bits würden in ihrer Darstellung als „binäre" Nullen und Einsen eine Dualzahl bilden.

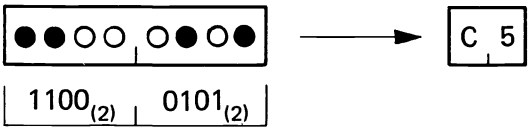


Abb. 54

Insgesamt sind 16 verschiedene vierstellige Dualzahlen möglich (Abb. 55), die jeweils einen dezimalen Wert zwischen 0 und 15 verkörpern. Ihre Kennzeichnung soll durch ein einstelliges Symbol erfolgen (10, 11, 12, 14, 14, 15 wären zweistellig), was durch die Verwendung hexadezimaler *) Ziffernsymbole 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E und F ermöglich wird.
Um eventuellen Mißverständnissen vorzubeugen, sei ausdrücklich darauf hingewiesen, daß ein „F" als Charakter (Bitkombination 1100 0110) etwas anderes repräsentiert als ein hexadezimales „F" (Bitkombination 1111).
Dem „F" ist natürlich nicht anzusehen, ob es nun ein Charakter oder ein hexadezimales Symbol sein soll; deshalb ist immer der ausdrückliche Hinweis auf eine byte- oder halbbyteweise *) Darstellung erforderlich.

mögliche Bitkombinationen innerhalb eines Halbbytes	hexadezimales Ziffernsymbol	dezimaler Wert entsprechend der dualen Darstellung
0000	0	0
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7
1000	8	8
1001	9	9
1010	A	10
1011	B	11
1100	C	12
1101	D	13
1110	E	14
1111	F	15

Abb. 55

* hexadezimal = 16

* Die halbbyteweise Darstellung wird, wie aus Kap. 12 hervorgeht, zur Kennzeichnung sogenannter gepackter Zahlenfelder verwendet.

In Abb. 56 ist der Teil eines Hauptspeichers wiedergegeben, in dem der Ausspruch

GUT GEBRUELLT, LOEWE!

gespeichert ist. Dieser Darstellung ist zu entnehmen, wie sich diese Information entsprechend den geschilderten drei Möglichkeiten darstellen läßt (zur Nachprüfung empfehlen sich die Angaben der Tabellen nach Abb. 20 und Abb. 55).

G	U	T		G	E	B	R
C 7	E 4	E 3	4 0	C 7	C 5	C 2	D 9
1100 0111	1110 0100	1110 0011	0100 0000	1100 0111	1100 0101	1100 0010	1101 1001

U	E	L	L	T	,		L
E 4	C 5	D 3	D 3	E 3	6 B	4 0	D 3
1110 0100	1100 0101	1101 0011	1101 0011	1110 0011	0110 1011	0100 0000	1101 0011

O	E	W	E	!	byteweise Darstellung
D 6	C 5	E 6	C 5	5 A	halbbyteweise Darstellung
1101 0110	1100 0101	1110 0110	1100 0101	0101 1010	bitweise Darstellung

Abb. 56

12. Dezimale und duale Rechenoperation

Mit dem Wort Computer verbindet sich unwillkürlich die Vorstellung von phantastisch hohen elektronischen Rechengeschwindigkeiten. Und fürwahr, manche Computertyen können mehrere Millionen Additionen pro Sekunde ausführen. Damit ist jeder Kopfrechner weit in den Schatten gestellt.
Aus Kapitel 10 ist bekannt, daß Zahlen in verschiedenen Zahlensystemen dargestellt werden können. Während das Dezimalsystem über insgesamt zehn mögliche Ziffernsymbole (0, 1, 2, ... 9) verfügt, begnügt sich das Dualsystem mit nur zwei Ziffernsymbolen (0 für O, 1 für L). Dem Aufbau der Dezimal- und Dualzahlen liegt die Stellenschreibweise zugrunde. Jeder Stelle ist eine bestimmte Wertstufe zugeordnet. Im Dezimalsystem werden die Wertstufen durch steigende Potenzen zur Basis 10 (10^0, 10^1, 10^2, 10^3, . . .) im Dualsystem durch steigende Potenzen zur Basis 2 (2^0, 2^1, 2^2, 2^3, . . .) gebildet – (Abb. 57).

	10^3	10^2	10^1	10^0	
	1	**9**	**7**	**0**	**Dezimalzahl**

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	
1	**0**	**1**	**1**	**0**	**0**	**0**	**1** $_{(2)}$	**Dualzahl**

Abb. 57

Ganz rechts steht die niedrigste, ganz links die höchste Wertstufe. Aufgrund der bei Speicherzellen möglichen Schaltzustände (ein oder aus) ist es recht einfach, eine Dualzahl im Hauptspeicher darzustellen. Dabei wird das Ziffernsymbol 1 (für L) durch eine eingeschaltete, das Ziffernsymbol O (für 0) durch eine ausgeschaltete Speicherzelle repräsentiert (Abb. 58).

● eingeschaltete Speicherzelle
○ ausgeschaltete Speicherzelle

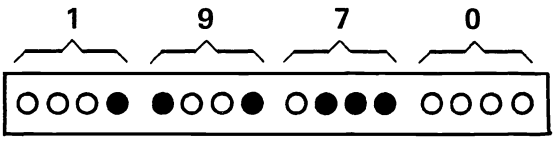
Darstellung der achtstelligen Dualzahl 1011 0001 in acht nebeneinanderliegenden Speicherzellen

Abb. 58

Bitkombination innerhalb eines Halbbytes	Dezimales Ziffernsymbol (entspricht dem hexadezimalen Symbol)
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9

Abb. 59

Etwas schwieriger gestaltet sich allerdings die Darstellung von Dezimalzahlen. Man unterscheidet dabei eine ungepackte von der gepackten Form. Rechenoperationen setzen immer die gepackte Form voraus, bei der für jede Stelle der Dezimalzahl vier Speicherzellen eines Halbtypes zur Verfügung stehen. In diesen vier Speicherzellen wird die Ziffer 0, 1, 2, 3, . . . oder 9 verschlüsselt. Im Falle einer Null sind zum Beispiel alle vier Speicherzellen ausgeschaltet. Die Tabelle nach Abb. 59 gibt über die Verschlüsselung aller zehn Ziffernsymbole Auskunft. Die Dezimalzahl 1970 würde demnach im Hauptspeicher wie folgt verschlüsselt werden (Abb. 60):



0 0 0 1	1 0 0 1	0 1 1 1	0 0 0 0	bitweise Darstellung
1	9	7	0	hexadezimale Darstellung

Abb. 60

Ein Vergleich der Abb. 58 und Abb. 60 zeigt, daß eine Dualzahlenstelle im Hauptspeicher eine einzige, eine Dezimalstelle hingegen vier Speicherzellen (=1/2 Byte) erfordert. Der Wert jedes dezimalen Ziffernsymboles wird innerhalb eines Halbbytes dual dargestellt.
Alle diese Betrachtungen haben natürlich einen recht praktischen Hintergrund. Sie dienen nämlich der Vorbereitung, endlich Näheres über die „Rechenkünste" des Computers zu erfahren. Und um es gleich vorwegzunehmen, der Computer kann sowohl mit Dezimal- als auch mit Dualzahlen prächtig umgehen. Er kann diese addieren, subtrahieren, multiplizieren und dividieren. Für das Rechnen mit Dezimalzahlen stehen die Instruktionen

AP (**A**ddiere ge**p**ackt)
SP (**S**ubtrahiere ge**p**ackt)
MP (**M**ultipliziere ge**p**ackt)
DP (**D**ividiere ge**p**ackt)

für das Rechnen mit Dualzahlen die Instruktionen

AR (**A**ddiere **R**egister)
SR (**S**ubtrahiere **R**egister)
MR (**M**ultipliziere **R**egister)
DR (**D**ividiere **R**egister)

zur Verfügung. Eine dezimale Rechenoperation führt immer zu einem dezimalen, eine duale Rechenoperation immer zu einem dualen Ergebnis. Nach den vorstehenden Bemerkungen ist es leicht, auf solche Instruktionen näher einzugehen.

Dezimales Rechnen

Wieder soll ein kleines Beispiel zur Erläuterung dienen, was beim dezimalen Rechnen alles zu berücksichtigen ist. In dem folgenden

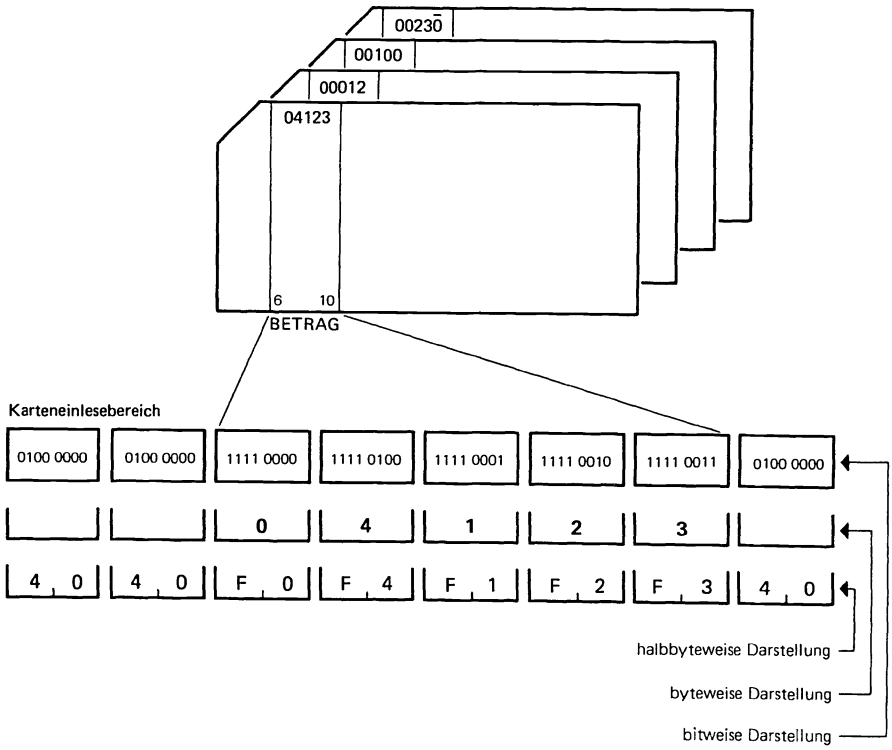


Abb. 61

kann man genau die einzelnen Schritte von einer Zahleneingabe in den Computer bis zur Ausgabe des gewünschten Ergebnisses verfolgen.
Die Aufgabe bezieht sich auf die Addition von Zahlen aus Lochkarten. Die Anzahl der Lochkarten ist unbestimmt; in Abb. 61 sind vier von ihnen abgebildet. In jeder Lochkarte ist eine Zahl in den Spalten 6 bis 10 rechtsbündig abgelocht. Rechtsbündig bedeutet, daß die Einerstelle der Zahl immer in der rechtesten Stelle (in diesem Fall der Spalte 10) des Lochkartenfeldes steht. Links freibleibende Stellen sind mit führenden Nullen ausgefüllt.
Sämtliche Lochkarten werden nacheinander eingelesen. Jedesmal wird die in ihnen enthaltene Zahl in ein bestimmtes zu definierendes Hauptspeicherfeld addiert. Nach Verarbeitung aller Lochkarten enthält dieses Feld die Gesamtsumme aller Zahlen, welche dann in eine sogenannte „Endsummenkarte" in die Spalten 16 bis 20 auszustanzen ist. Nach dem Einlesen der ersten Lochkarte hat der dem Zahlenfeld entsprechende Teil des Karteneinlesebereiches den in Abb. 61 angegebenen Inhalt. Jedes Byte enthält eine Ziffer in einer ungepackt verschlüsselten Form. Um darüber etwas Genaueres aussagen zu können, stellt man sich jedes Byte in ein rechtes und linkes Halbbyte aufgeteilt vor (Abb. 62).

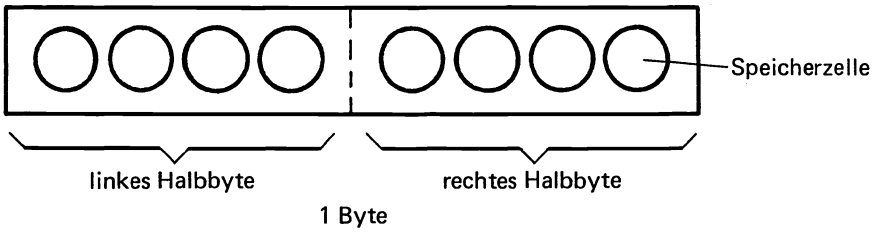


Abb. 62

Die linken Halbbytes in Abb. 61 zeigen durchweg binäre Einsen (hexadezimale F's), während in den vier Bits der rechten Halbbytes die eigentlichen Ziffernwerte nach Dualgesichtspunkten gespeichert sind.
Schließlich ist auch noch das Vorzeichen (positiv oder negativ) einer Zahl auf irgendeine Weise zu berücksichtigen. Zu diesem Zweck kann in der Einerstelle, also in der rechtesten Spalte des Zahlenfeldes eine zusätzliche Lochung vorgesehen sein (Abb. 63). Eine

Vorzeichen	zusätzliche Lochung in der Einerstelle des Zahlenfeldes	Bitkombination	hexadezimale Darstellung
+	12	1100	C
–	11	1101	D
kein Vorzeichen		1111	F

Abb. 63

12er-Lochung bedeutet ein positives, eine 11er-Lochung ein negatives Vorzeichen. Eine vorzeichenlose Zahl (ohne 12er bzw. 11er-Lochung) erkennt der Computer als quasi positiv an. Im Karteneinlesebereich (Abb. 61) erscheint das Vorzeichen des Zahlenfeldes immer in dem linken Halbbyte der Einerstelle.
Ein Blick auf Abb. 61 läßt erkennen, daß die vierte Lochkarte die negative Zahl – 00230 enthält, die wie folgt (Abb. 64) abgelocht ist:

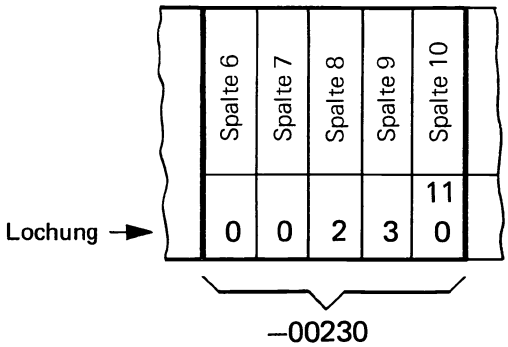


Abb. 64

Das im Karteneinlesebereich vorliegende ungepackte Zahlenfeld ist jedoch für dezimale Rechenoperationen nicht zugänglich. Es muß deshalb zunächst in ein gepacktes Zahlenfeld überführt werden, in der die lästigen Zonen (hexadezimale F's) entfallen und in der nur die eigentlichen Ziffern auftreten (Abb. 65).
Erst mit einem solchen gepackten Zahlenfeld können dezimale Rechenoperationen ausgeführt werden. Es ist selbstverständlich, daß dabei auch das Vorzeichen Berücksichtigung findet.
Das Vorzeichen steht immer im rechtesten Halbbyte des gepackten Feldes.

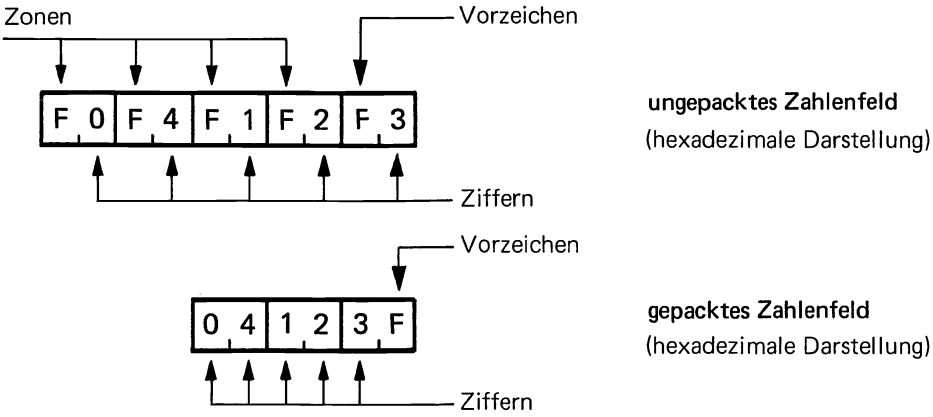


Abb. 65

Für die Überführung eines ungepackten in ein gepacktes Zahlenfeld steht dem Programmierer die PACK-Instruktion zur Verfügung. Als deren erster Operand (vor dem Komma) muß der Name des Feldes genannt werden, das für die gepackte Dezimalzahl vorgesehen ist; als zweiter Operand (nach dem Komma) muß der Name des ungepackten Feldes erscheinen.
Im Programm für das Ausführungsbeispiel Nr. 5 trägt der Karteneinlesebereich den Namen EINLES, und das (3 Byte große) Feld für die gepackte Zahl den Namen REBE. Das ungepackte Zahlenfeld reicht von der sechsten bis zehnten Stelle des Karteneinlesebereiches; die symbolische Bezeichnung für das ungepackte Zahlenfeld lautet demnach EINLES+5(5). Unter Berücksichtigung dieser Operanden läßt sich die vollständige PACK-Instruktion wie folgt niederschreiben:

```
PACK    REFE,EINLES+5(5)
```

In Abb. 66 ist schematisch angedeutet, wie sich die Ausführung dieser Instruktion auswirkt. Das Vorzeichen des ungepackten Feldes gelangt, der ausgezogenen Linie folgend, in das Vorzeichenhalbbyte des gepackten Feldes. Das gepackte Feld wird anschließend mit dem Inhalt der Ziffernhalbbytes des ungepackten Zahlenfeldes von rechts nach links (gemäß den gestrichelten Linien) aufgefüllt. Auf diese Art und Weise gelangt man zu einer gepackten Dezimalzahl, die sich mit einer anderen durch dezimale Rechenoperationen (AP, SP, MP, DP) verknüpfen läßt.
Bei der Anwendung einer PACK-Instruktion dürfen nur Zahlenwerte und keine alphabetischen Angaben gepackt werden. Denn nur so ist gewährleistet, daß in dem gepackten Feld neben dem Vorzeichen nur die Ziffern 0 bis 9 (in hexadezimaler Darstellung) auftreten können. Und dieser Umstand ist sehr wichtig, denn mit anderen Größen würde und könnte der Computer nicht rechnen. Mit der Ausnahme einiger Bemerkungen steht nun der Lösung des Ausführungsbeispieles Nr. 5 kaum noch etwas im Wege. Die Entwicklung der einzelnen Programmschritte läßt sich am einfachsten anhand des fertigen Blockdiagrammes (Abb. 67) und des Zeilen-Schemas (Abb. 68) verfolgen.

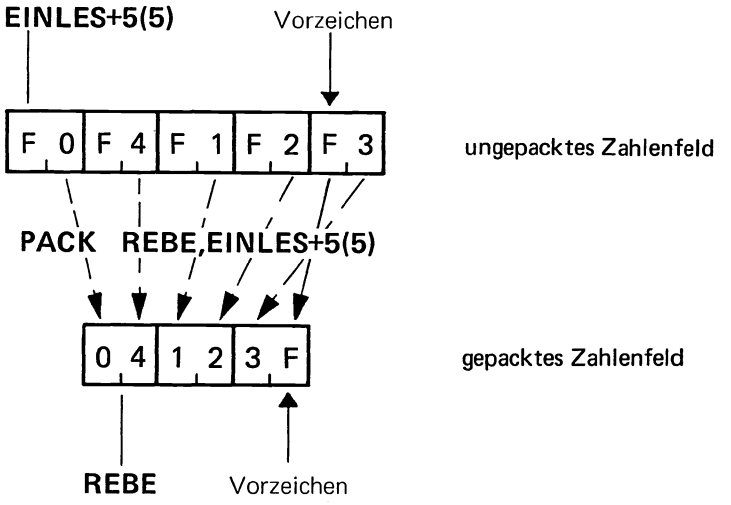
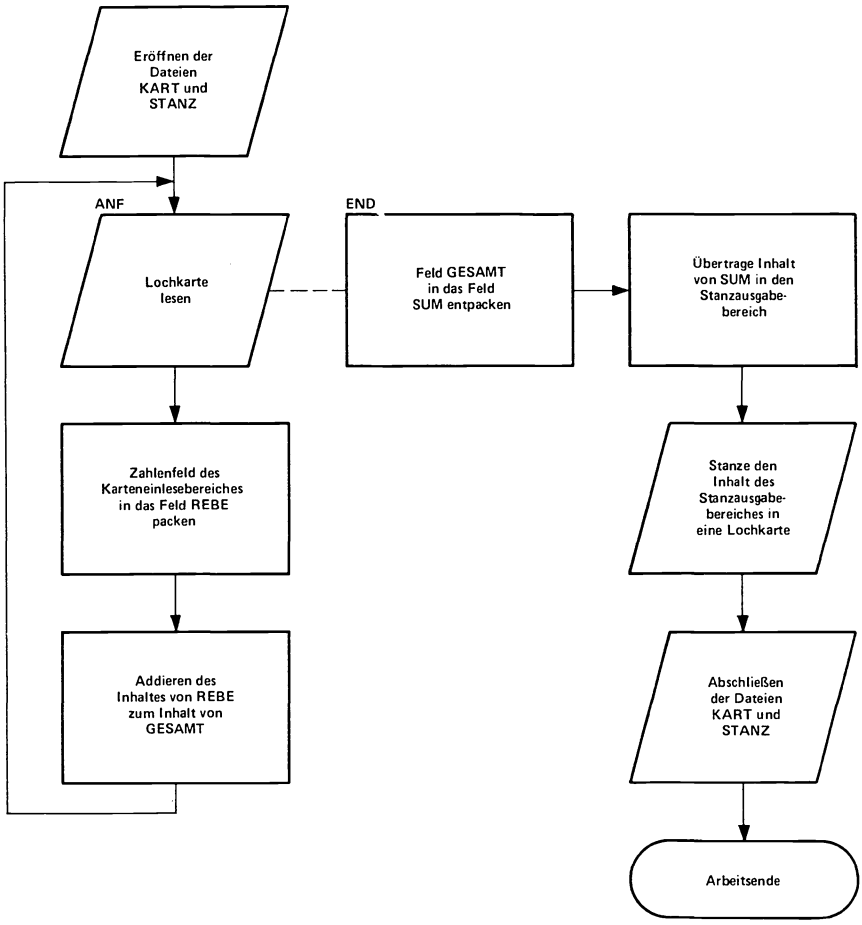


Abb. 66



(Die verwendeten symbolischen Bezeichnungen gehen aus Abb. 68 hervor) **Abb. 67**

Nach dem Einlesen einer Lochkarte wird der entsprechende Betrag aus dem Karteneinlesebereich in das Rechenfeld REBE gepackt und von dort aus durch die Instruktion

```
AP      GESAMT,REBE
```

in das Rechenfeld GESAMT addiert. Das Ergebnis dieser Addition erscheint immer im ersten Operanden, also im Feld GESAMT. Bevor diese Instruktion das allererstemal ausgeführt wird, muß das Feld GESAMT den Inhalt Null aufweisen. Es wäre falsch, ihm per Definition den Inhalt Blank zuzuschreiben. Denn dann blieben die Anforderungen, denen dieses Rechenfeld genügen muß, unerfüllt. Diese Anforderungen schreiben ein gültiges Vorzeichen und für die übrigen Halbbytes einen Inhalt von jeweils einer Null (in hexadezimaler Darstellung) vor. (Abb. 69).

SIEMENS

Assembler

Programm _____ Problem Blatt

Bearbeiter _____ Datum _____

Name	Operation	Operanden und Bemerkungen
		Dateibeschreibungen
		Für die Lochkarten auf der Kartenfühleinheit
		Name der Datei KART
		Name des Karten-einlesebereiches EINLES
		Art der Datei INPUT
		EOF-Merkmal END
		Für die Lochkarten auf der Stanzeinheit
		Name der Datei STANZ
		Name des Stanz-ausgabebereiches AUSSTA
		Art der Datei OUTPUT
	OPEN	KART,STANZ
ANF	GET	KART
	PACK	REBE,EINLES+5(5)
	AP	GESAMT,REBE
	B	ANF
END	UNPK	SUM,GESAMT
	MVC	AUSSTA+15(5),SUM
	PUT	STANZ
	CLOSE	KART,STANZ
	EOJ	
**DEFINITIONEN		
EINLES	DC	CL80' '
AUSSTA	DC	CL80' '
REBE	DC	CL3' '
GESAMT	DC	X'00000F'
SUM	DC	CL5' '

Spalten 73—78 siehe Kopfzeile

Mehrfachlochungen sind in derselben Zeile als Bemerkung unter Angabe der Lochkombination zu erläutern.
Die Spalte 80 ist nur bei Einschiebungen auszufüllen. Es sollte mit „A" begonnen werden. Einschübe können in die letzten sechs Zeilen geschrieben werden.

Abb. 68

Man kann zu diesem Zweck das Feld GESAMT halbbyteweise wie folgt definieren:

```
GESAMT   DC   X'00000F'
```

Das X vor dem ersten Hochkomma deutet dem Computer eine nachfolgende hexadezimale Konstante an, deren Inhalt halbbyteweise anzugeben ist. Durch die Definitonsanweisung GESAMT DC X'00000F' erhält das Feld GESAMT den in Abb. 69 angege-

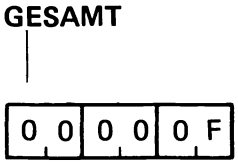


Abb. 69

benen Inhalt. In dieser Form ist es einer dezimalen Rechenoperation zugänglich.
In Abb. 70 wird gezeigt, welchen Inhalt die Rechenfelder REBE und GESAMT vor der erstmaligen Ausführung der Instruktion

```
AP      GESAMT,REBE
```

haben. Daß bei der Ergebnisbildung plötzlich ein hexadezimales C und kein F als positives Vorzeichen auftritt, soll nicht weiter verwundern, denn diese Vorzeichen sind einander gleichwertig. Aus der Tabelle nach Abb. 71 gehen sämtliche gültigen Vorzeichen hervor, die in gepackten Rechenfeldern überhaupt auftreten können.

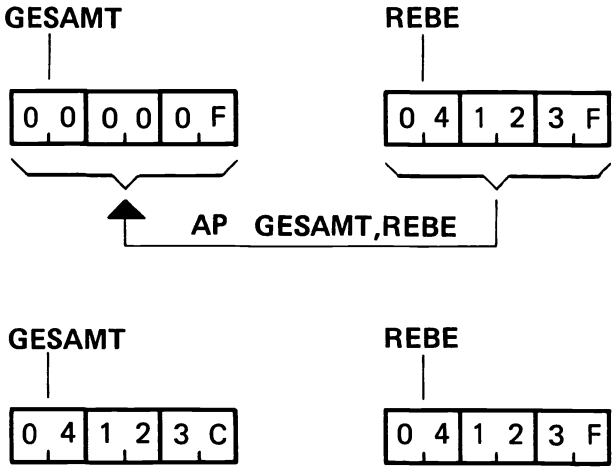


Abb. 70

Nach dem Einlesen der zweiten Lochkarte (Abb. 61) wird zum Inhalt 04123 des Feldes GESAMT der Wert 00012 addiert. Bei der dritten Lochkarte kommte dann der Wert 00100 und bei der vierten Lochkarte der Wert -00230 hinzu, so daß schließlich im Feld GESAMT die Endsumme 04005 der Zahlen aus allen Lochkarten erscheint. Bei der

Vorzeichen	Bitkombination	hexadezimales Symbol
+	1100	C
–	1101	D
(+)	1010	A
(+)	1110	E
(+)	1111	F
(–)	1011	B

Abb. 71

vierten Lochkarte ist der zu addierende Wert negativ. Der Computer muß in diesem Falle das negative Vorzeichen berücksichtigen. Alle Zweifel, daß er dies eventuell unterlassen könnte, bestehen zu Unrecht; denn mit der AP-Instruktion können sowohl positive als auch negative Werte vorzeichengetreu addiert werden.
So ergäbe zum Beispiel:

	1	addiert zu		1	=	+	2
	2	addiert zu	+	1	=	+	3
	2	addiert zu	–	1	=	+	1
+	1	addiert zu		6	=	+	7
+	3	addiert zu	+	5	=	+	8
+	4	addiert zu	–	9	=	–	5
–	7	addiert zu		3	=	–	4
–	6	addiert zu	+	3	=	–	3
–	4	addiert zu	–	3	=	–	7

Es ließe sich leicht nachprüfen, daß die Addition der Zahlen aus den Lochkarten (Abb. 61) zu der richtigen Endsumme +04005 führen muß. Dieser Wert ist allerdings nur in gepackter Form im Feld GESAMT verfügbar (Abb. 72). Doch diese Form ist für die geforderte Datenausgabe noch völlig ungeeignet. Es ist ja bereits bekannt, daß das Ausdrucken des Inhaltes eines Hauptspeicherfeldes immer nur byteweise erfolgt. Im übertragenen Sinne gilt dies natürlich auch für das Ausstanzen von Angaben in Lochkarten. Dabei ist jedem Byte des Stanzausgabebereiches eine Lochkartenspalte zugeordnet. Würde man den gepackten Inhalt des Feldes GESAMT unverändert in den Stanzausgabebereich übertragen und ausgeben, so ergäbe sich ein völlig falsches Ergebnis, das der byteweisen Darstellung in Abb. 72 entsprechen würde (für eine hexadezimale Inhaltsangabe 04 bzw. 00 gibt es keine zugeordneten Zeichen; punktierte Inhaltsangabe). Es ist also unbedingt erforderlich, den Inhalt des gepackten Feldes GESAMT in eine für das

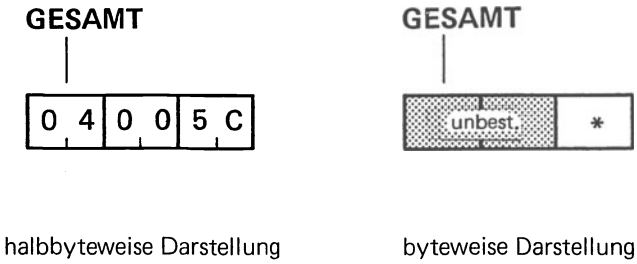


Abb. 72

Ausstanzen bzw. Ausdrucken des Ergebnisses geeignete ungepackte Form zu überführen. Diesem Zweck dient eine UNPACK-Instruktion. Mit deren Hilfe ist es möglich, aus einem gepackten ein ungepacktes Zahlenfeld zu erzeugen. Laut Abb. 73 wurde zur Aufnahme des ungepackten Ergebnisses das fünf Byte große Feld SUM definiert.
Bei der Ausführung der Instruktion

UNPK SUM,GESAMT

(Abb. 73) gelangt das Vorzeichen C gemäß der ausgezogenen Linie an den dafür vorgeschriebenen Platz (linkes Halbbyte des rechtesten Bytes). Der Inhalt der Ziffernhalbbytes des gepackten Feldes folgt den gestrichelten Linien in das Feld SUM, wobei dessen Zonenhalbbytes mit hexadezimalen F's aufgefüllt werden.
Das Feld SUM enthält jetzt das Ergebnis in einer für das Ausstanzen geeigneten ungepackten Form (entsprechend der byteweisen Darstellung). Im Programm ist also nur noch der Inhalt von SUM in die vorgeschriebenen Stellen 16 bis 20 des Stanzausgabebereiches *) zu übertragen und dann kann bereits der Stanzbefehl erteilt werden. Nach

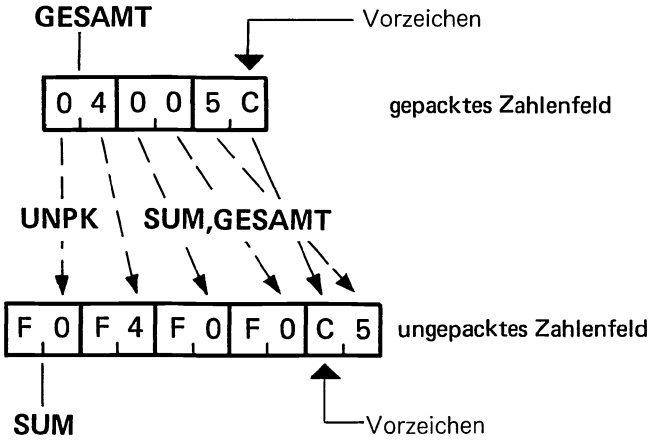
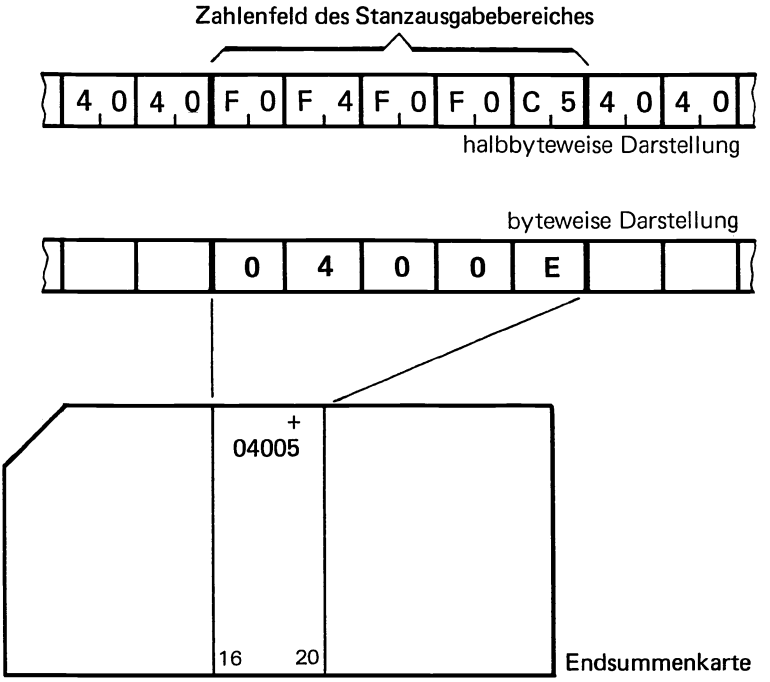


Abb. 73

* Der Stanzausgabebereich AUSSTA wurde entsprechend den 80 Stellen einer Lochkarte in der Größe von 80 Bytes definiert.



Bitkombination	Darstellung halbbyteweise	Darstellung byteweise	Lochung
0100 0000	4 0	(blank)	keine
1100 0101	C 5	E	12,5
1111 0100	F 4	4	4
1111 0000	F 0	0	0

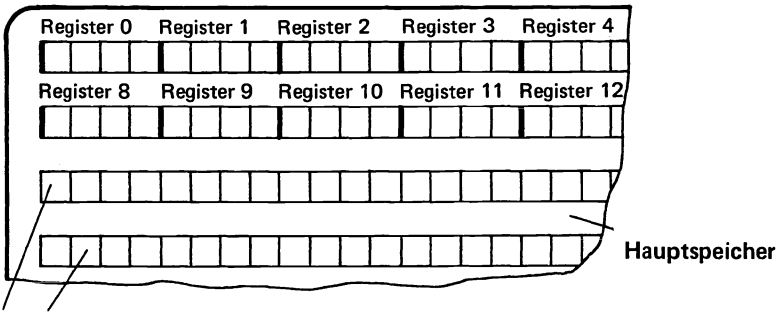
Abb. 74

dessen Ausführung erscheint das Ergebnis in der Lochkarte für die Endsumme (Abb. 74). Das Zeichen E stellt sich in Spalte 20 als Fünfer-Ziffernlochung und als 12er-Lochung für ein positives Vorzeichen dar (Abb. 74).

Duales (binäres) Rechnen

In unserem Alltagsleben hat sich das Dezimalsystem so gut bewährt, daß wohl niemand mehr auf den Gedanken käme, sich anderer Zahlensysteme zu bedienen. Anderfalls wäre zum Beispiel das Finanzamt bestimmt mehr als erstaunt, wenn auf Steuererklärungen plötzlich Dualzahlen erscheinen würden. Etwas anders liegen die Verhältnisse allerdings

in der Computertechnik. Hier drängen die technischen Gegebenheiten förmlich zum Dualsystem. Denn nichts ist einfacher, als eine Dualzahl mit Hilfe ein- oder ausgeschalteter Speicherzellen darzustellen (Abb. 58).
In diesem Zusammenhang ist es natürlich angebracht, auch etwas über den Umgang des Computers mit Dualzahlen zu erfahren: Wie dieser mit ihnen rechnet und welche Vorteile damit verbunden sind.
Zuvor muß jedoch die Kenntnis des Hauptspeichers noch etwas erweitert werden. Die Vorstellung aneinandergereihter Speicherstellen bleibt dabei selbstverständlich erhalten. Es wird lediglich der Begriff des Registers eingeführt. Allzuoft tauchen besonders in der Programmierung die Bezeichnungen Register 0, Register 1, Register 2, ... Register 15 auf.



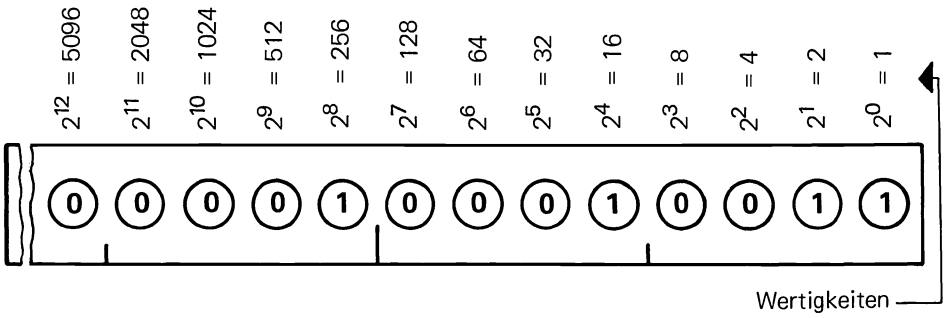
Nur in diese registerungebundenen Speicherstellen wird ein Programm geladen

Abb. 75

Ohne die Einrichtung der Register wäre eine elegante Programmierung fast undenkbar (Kapitel 16).
Doch es hat wohl wenig Zweck, über ein Register zu sprechen, bevor man nicht weiß, was es nun eigentlich darstellt. Die Antwort fällt sehr einfach aus, denn ein Register ist nichts weiter als ein Hauptspeicherfeld mit einem festen Platz und einer bestimmten Größe (zum Beispiel 4 Bytes).
Sämtliche Register sind im „oberen" Teil des Hauptspeichers angeordnet (Abb. 75). Der Computer kann sich auf jedes dieser Register einstellen, sobald er durch Angabe der Registernummer in einer Instruktion dazu aufgefordert wird.
Soll beispielsweise der Inhalt des Registers 6 zu dem des Registers 5 addiert werden, so ist dazu lediglich die Ausführung der Instruktion

```
AR      5,6
```

erforderlich (das Ergebnis erscheint immer im ersten Operanden, in diesem Falle also im Register 5). Als Registerinhalt können ganz im Gegensatz zu gewöhnlichen Hauptspeicherplätzen nur Zahlen auftreten, Zahlen, die immer dual verschlüsselt sind.
Dabei wird jeder der 32 Speicherzellen eines vier Byte großen Registers eine von rechts nach links steigende Wertigkeit zugeordnet ($2^0=1$; $2^1=2$; $2^2=4$, $2^3=8$, $2^4=16$, $2^5=32$; usw). Ein eingeschaltete Speicherzelle repräsentiert wieder eine binäre Eins, eine ausgeschalteter eine binäre Null. Geht es darum, den dezimalen Wert einer im Register gespeicherten Dualzahl zu erfassen, so müßte man jeweils den Binärwert mit der zugehörigen Wertigkeit multiplizieren und alle Teilprodukte aufaddieren (Abb. 76).



Register (4 Bytes = 32 Bits)

	1 x	1 =	1
+	1 x	2 =	2
+	0 x	4 =	0
+	0 x	8 =	0
+	1 x	16 =	16
+	0 x	32 =	0
+	0 x	64 =	0
+	0 x	128 =	0
+	1 x	256 =	256
+	0 x	512 =	0
+	.....		
			275

Abb. 76

Es ist allerdings immer noch die Frage offen, auf welchem Wege Zahlen in Register gelangen. Denn erst dann sind sie dualen Rechenoperationen zugänglich. Hier sind dem Programmierer zwei prinzipielle Möglichkeiten gegeben:

Einmal kann er mit einer CVB-Instruktion (convert to binary = umwandeln in binär bzw. dual) den Inhalt eines gepackten Zahlenfeldes in ein benanntes Register überführen *)

```
CVB    6,GESAMT
```

Name des Feldes mit der gepackten Dezimalzahl

Registernummer

und zum anderen bietet sich die Möglichkeit, mit Hilfe einer LA-Instruktion (load address = Lade Adresse) einen vorgegebenen Wert in ein Register zu laden.

* Bei Ausführung der CVB-Instruktion wird eine gepackte Dezimalzahl dual verschlüsselt in ein Register überführt.

```
LA      8,56
```

diese Dezimalzahl wird dual verschlüsselt in das Register 8 geladen

Der Vorteil aller dualer Rechenoperationen (AR, SR, MR und DR) liegt einesteils in deren enorm kurzer Ausführungszeit. Eine AR-Instruktion läuft nun eben einmal bedeutend schneller als eine AP-Instruktion ab. Als nachteilig mag wohl zunächst noch empfunden werden, daß duale Rechenoperationen immer zu einem dualen Ergebnis führen, welches jedoch in dieser Form zum Ausdrucken auf einer Liste höchst ungeeignet ist. Denn jeder Kaufmann oder Wissenschaftler wünscht stets nur Dezimalzahlen zu lesen. Um dies zu ermöglichen, muß auf die CVD-Instruktion (convert to decimal = umwandeln in dezimal) zurückgegriffen werden. Diese Instruktion gestattet die Umwandlung eines dualen Registerstandes in eine gepackte Dezimalzahl

```
CVD     6,GESAMT
```

Name des Feldes für die gepackte Dezimalzahl

Registernummer

Sobald das Ergebnis in gepackter Form vorliegt, bereiten die weiteren Schritte, das Entpacken und die Übertragung in den Ausgabebereich und das Ausdrucken keine Schwierigkeiten mehr. Dieser Weg erscheint gewiß recht umständlich. Denn es mag bestimmt noch nicht einleuchten, weshalb man sich nicht grundsätzlich mit dezimalen Rechenoperationen begnügen kann. Denn so könnte man sich das duale Rechnen im Zusammenhang mit der CVB- bzw. CVD-Instruktion ersparen.
Und doch birgt das duale Rechnen in den Registern ganz wesentliche Vorteile für die Programmierung in sich. Diese offenbaren sich aber erst in einem ganz anderen Zusammenhang, wenn es nämlich darum geht, mit Speicherplatzadressen zu rechnen; wenn sich zum Beispiel die Forderung ergibt, zu der absoluten (echten Adresse) eines symbolischen Feldes TAB den Wert 56 zu addieren.
Es sei vorausbemerkt, daß sämtliche Speicherplatzadressen in Wirklichkeit dual verschlüsselt sind. Außerdem ist der Computer in der Lage, auf Anforderung die echte (absolute) Adresse eines symbolisch benannten Feldes in ein Register zu laden. So würde die LA-Instruktion (Load address = Lade Adresse) in der Schreibweise

```
LA      9,TAB
```

Name des Feldes

Registernummer

bewirken, daß die Adresse (nicht der Inhalt) 192 des Feldes TAB (Abb. 77) in das Register 9 geladen wird. Aus Gründen der Vereinfachung sind in Abb. 77 die Speicheradressen ausnahmsweise nicht dual, sondern dezimal dargestellt (tatsächlich müßte also der Inhalt von Register 9 nicht 192, sondern 1100 0000 lauten).

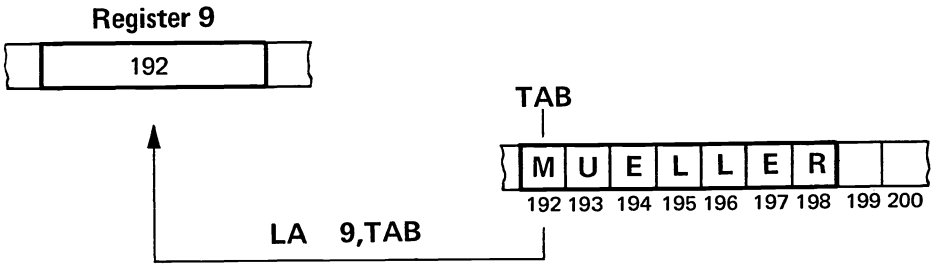


Abb. 77

Zu diesem Registerstand kann der Wert 56 auf zweierlei Wegen addiert werden.
Entweder lädt man zuvor die 56 in ein anderes Register, zum Beispiel in das Register 8, um anschließend den Inhalt beider Register 8 und 9 über die AR-Instruktion zu verknüpfen

```
LA    9,TAB
LA    8,56
AR    8,9
```

oder man benutzt die LA-Instruktion in einer besonderen Schreibweise zur Erhöhung des Inhaltes des Registers 8 um den Wert 56

```
LA    8,TAB
LA    8,56(8,0)
```

der Inhalt des Registers 8
wird um 56 erhöht

an dieser Stelle muß aus formellen Gründen eine Null stehen

In beiden Fällen würde das Ergebnis, die um 56 erhöhte Adresse des Feldes TAB als Dualzahl im Register 8 erscheinen. Und dieser Wert könnte dem Computer als Hinweis dienen, sich auf einen Speicherplatz mit dieser Adresse einzustellen.
Eine derartige Einbeziehung einer echten (absoluten) Adresse in eine Rechenoperation wäre beim dezimalen Rechnen undenkbar. Häufig muß der Programmierer aber von einer solchen Möglichkeiten Gebrauch machen, wie es zum Beispiel aus Kapitel 16 hervorgeht.
Abschließend ist in Abb. 78 gezeigt, wie sich die Zahl +275 als ungepackte und gepackte Dezimalzahl, sowie als Dualzahl darstellt.

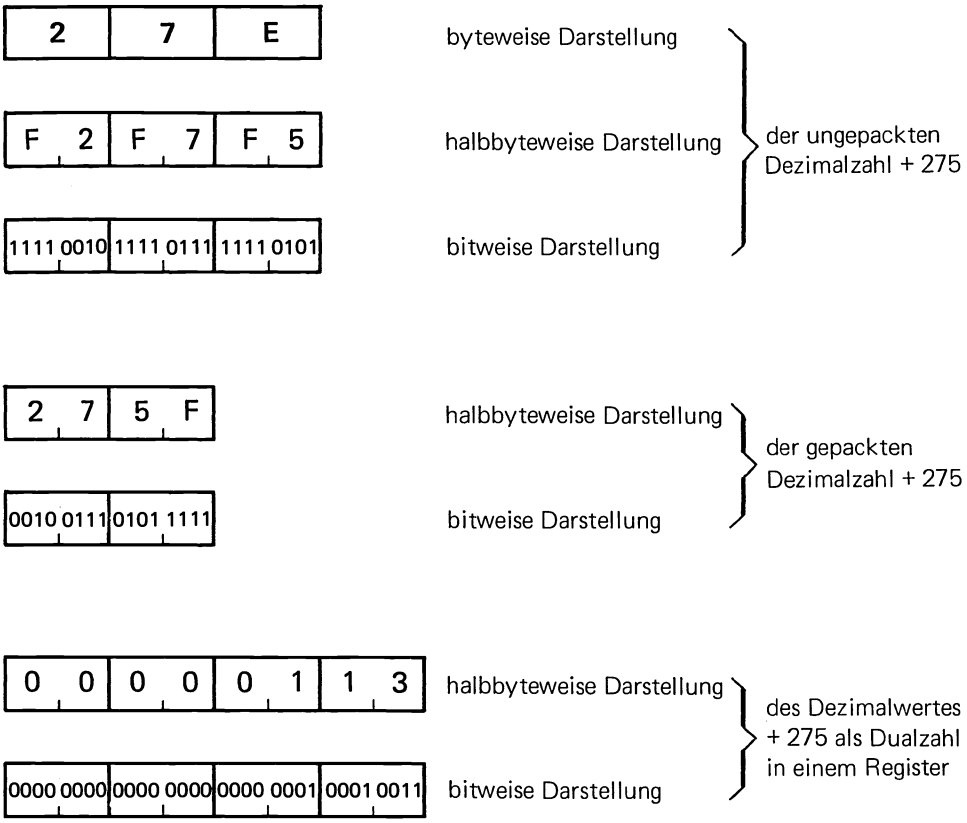
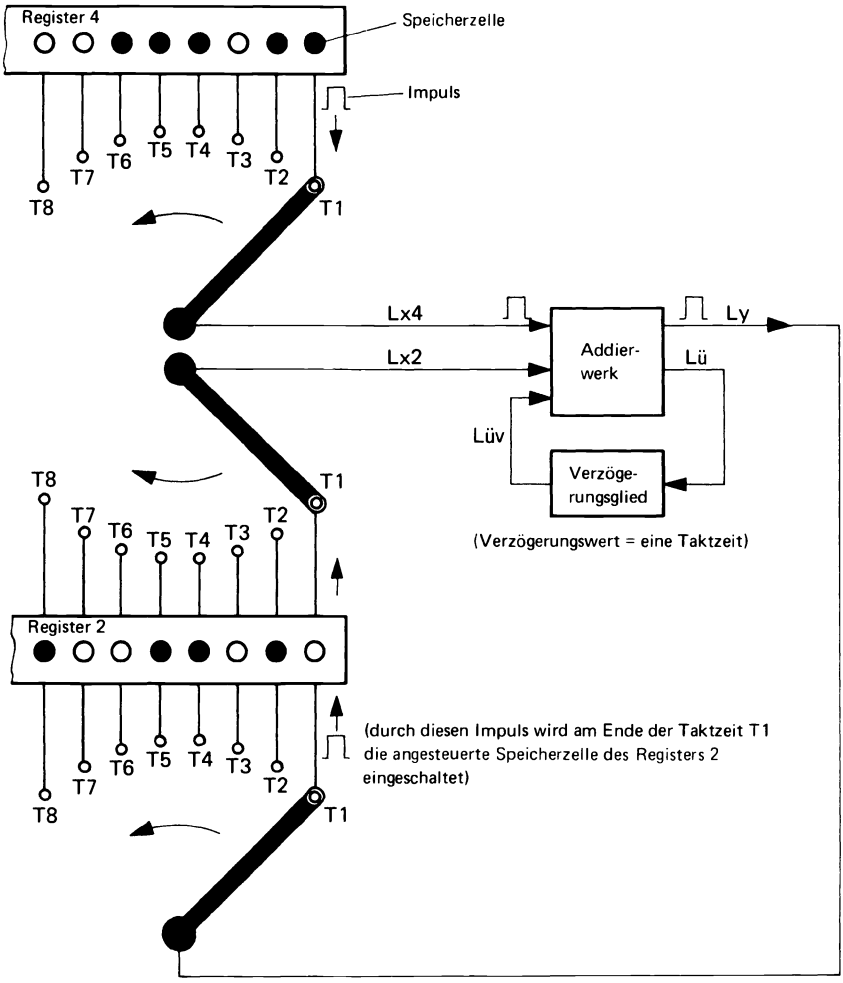

2 7 E
byteweise Darstellung
F 2 F 7 F 5
halbbyteweise Darstellung
1111 0010 1111 0111 1111 0101
bitweise Darstellung
der ungepackten Dezimalzahl + 275
2 7 5 F
halbbyteweise Darstellung
0010 0111 0101 1111
bitweise Darstellung
der gepackten Dezimalzahl + 275
0 0 0 0 0 1 1 3
halbbyteweise Darstellung
0000 0000 0000 0000 0000 0001 0001 0011
bitweise Darstellung
des Dezimalwertes + 275 als Dualzahl in einem Register

Abb. 78

13. Das Dualzahlenaddierwerk des Computers

Nach allem, was bisher über den Computer gesagt wurde, blieb eines noch ungeklärt: Die Arbeitsweise seiner elektronischen Schaltkreise. In diesem Kapitel soll deshalb ein kleiner Ausflug in die Computer-Elektronik unternommen werden; dabei wird ein Dualzahlen-Addierwerk in einer für den Laien verständlichen Form behandelt.
(Wer am Aufbau von Schaltkreisen weniger Interesse hat, kann dieses Kapitel ohne Nachteil für das Verständnis der folgenden überschlagen).



Prinzipdarstellung eines Additionsvorganges

Abb. 79

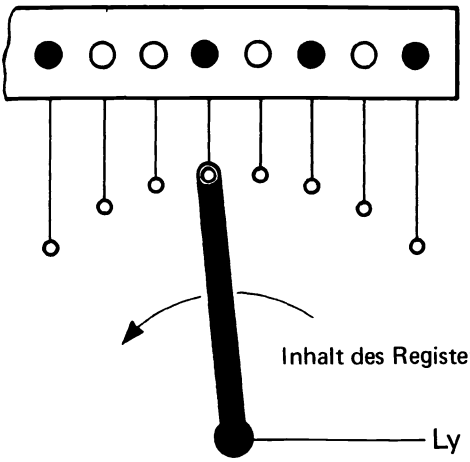


Abb. 80

Zur Erklärung der Funktionsweise dieses Addierwerkes soll die Instruktion

AR 2,4

dienen. Diese Instruktion bedingt die Addition des Inhaltes des Registers 4 zu dem des Registers 2 (dort erscheint auch das Ergebnis).
Zur Vereinfachung sind in Abb. 79 von jedem Register nur die letzten acht der insgesamt 32 Bits dargestellt. Es gilt weiterhin zu beachten, daß der Inhalt eines Registers immer als Dualzahl (Kapitel 12) zu interpretieren ist.
In dem Beispiel nach Abb. 79 hat das Register 4 einen Inhalt von 0011 1011 (dezimaler Wert 59), das Register 2 einen Inhalt von 1001 1010 (dezimaler Wert 154). Nach Ausführung der Instruktion

AR 2,4

würde das Ergebnis – die Summe 1101 0101 (dezimaler Wert 213) – im Register 2 stehen.
Der Additionsvorgang erfolgt schrittweise. Den Zeitpunkt eines jeden Schrittes bestimmt ein zentraler Taktgeber: der erste Takt tritt zum Zeitpunkt T1, der zweite zum Zeitpunkt T2, der dritte zum Zeitpunkt T3 usw. auf. In dieser Taktfolge werden die Leitungen Lx4 und Lx2 von Speicherzelle zu Speicherzelle des entsprechenden Registers fortgeschaltet (Abb. 79). Wenn sich die angesteuerte Speicherzelle des Registers 4 bzw. des Registers 2 im eingeschalteten Zustand („1") befindet, wird auf der Leitung Lx4 bzw. Lx2 ein elektrischer Impuls erzeugt und dem Addierwerk zugeführt. Andernfalls bleibt die Adressleitung impulsfrei.
Das Addierwerk kann taktweise Impulse auf den Leitungen Lx4 und Lx2 addieren. Das Ergebnis einer Impulsaddition wird noch während derselben Taktzeit in Form eines vorhandenen oder fehlenden Impulses auf der Leitung Ly gebildet. Über die verschiedenen Möglichkeiten, die bei einer solchen Impulsaddition auftreten können, gibt die folgende Aufstellung (Abb. 81) Auskunft:

	Zustand auf der Leitung Lx4	Zustand auf der Leitung Lx2	Zustand auf der Leitung Ly
a)	kein Impuls	kein Impuls	kein Impuls
b)	kein Impuls	Impuls	Impuls
c)	Impuls	kein Impuls	Impuls
d)	Impuls	Impuls	kein Impuls

Abb. 81

Der gleiche Sachverhalt läßt sich aber auch in einer Funktionstabelle nach Abb. 82 zusammenfassen. Die beiden Eingangsgrößen sind als x4 und x2, die Ausgangsgröße als y bezeichnet. Jede Größe kann einen oder auch keinen Impuls darstellen, was als binäre Eins bzw. Null gekennzeichnet wird.

x4	x2	y
0	0	0
0	1	1
1	0	1
1	1	0

Abb. 82

In dieser Funktionstabelle spiegeln sich die Additionsgesetze von Dualzahlen wider :

$$\begin{array}{r} 0_{(2)} \\ +\ 0_{(2)} \\ \hline =\ 0_{(2)} \end{array} \qquad \begin{array}{r} 0_{(2)} \\ +\ 1_{(2)} \\ \hline =\ 1_{(2)} \end{array} \qquad \begin{array}{r} 1_{(2)} \\ +\ 0_{(2)} \\ \hline =\ 1_{(2)} \end{array} \qquad \begin{array}{r} 1_{(2)} \\ +\ 1_{(2)} \\ \hline =\ 10_{(2)} \end{array}$$

diese 1 versteht sich als Übertrag.
($1_{(2)} + 1_{(2)} = 0_{(2)}$ plus Übertrag 1 für die nächsthöhere Wertstufe)

Würde man nach diesen Gesetzen die in den Registern 4 und 2 gespeicherten Dualzahlen addieren, so ergäbe sich folgendes Bild:

```
  0 0 1 1 1 0 1 1
+ 1 0 0 1 1 0 1 0
  ---------------
    1 1 1   1
  1 1 0 1 0 1 0 1
```

Vermerk des Übertrages

Richtung, in der die Addition stellenweise auszuführen ist

Dabei kann der Fall eintreten, daß zu jeweils zwei Dualwerten (0 und 0, 0 und 1, 1 und 0, 1 und 1) noch ein Übertrag 1 aus der vorausgehenden Wertstufe zu addieren ist. Dann ist es zweckmäßig, sich zunächst die beiden Dualwerte in einem Teilergebnis zusammengefaßt zu denken, welchem dann der Übertrag aufzuaddieren ist:

$0_{(2)}$	$0_{(2)}$	$1_{(2)}$	$1_{(2)}$	erster Dualwert
$+\ 0_{(2)}$	$+\ 1_{(2)}$	$+\ 0_{(2)}$	$+\ 1_{(2)}$	zweiter Dualwert
$0_{(2)}$	$1_{(2)}$	$1_{(2)}$	$10_{(2)}$	Teilergebnis
$+\ 1_{(2)}$	$+\ 1_{(2)}$	$+\ 1_{(2)}$	$+\ 1_{(2)}$	Übertrag aus der vorausgehenden Wertstufe
$=\ 1_{(2)}$	$=10_{(2)}$	$=10_{(2)}$	$=11_{(2)}$	Endergebnis

diese Einsen stellen den Übertrag für die nächsthöhere Wertstufe dar

Das Addierwerk führt die Dualzahlenaddition ebenfalls stellenweise aus, indem es auf entsprechende fehlende oder vorhandene elektrische Impulse anspricht.
Während des ersten Taktes wird die Addition für die niedrigste Dualzahlenstelle vollzogen. Selbstverständlich kann sich dabei auch ein Übertrag ergeben; falls dieser Fall eintritt, erscheint auf der Übertragsleitung Lü ein Impuls (Abb. 79).

Da dieser Übertragsimpuls aber erst in der nächsthöheren Wertstufe zu berücksichtigen ist, wird er um eine Taktzeit verzögert auf der Leitung Lüv dem Eingang des Addierwerkes zugeführt. (Der Impulsverzögerung dient eine spezielle elektronische Schaltung). Das Additionswerk arbeitet so schnell, daß unmittelbar nach dem Auftreten der Eingangsimpulse noch während derselben Taktzeit die Ausgangsimpulse zur Verfügung stehen. Über die Verwendung der Impulse auf der Leitung Ly wird zu einem späteren Zeitpunkt im Zusammenhang mit der Ergebnisbildung etwas ausgesagt.
Unter Berücksichtigung evtl. auftretender Übertragsimpulse erweitert sich die Darstellung nach Abb. 81 zu Abb. 83.

Zustand auf der Leitung Lx4	**Zustand auf der Leitung Lx2**	**Zustand auf der Leitung Ly**	**Zustand auf der Leitung Lü**
kein Impuls	**kein Impuls**	**kein Impuls**	**kein Impuls**
kein Impuls	**Impuls**	**Impuls**	**kein Impuls**
Impuls	**kein Impuls**	**Impuls**	**kein Impuls**
Impuls	**Impuls**	**kein Impuls**	**Impuls**

Abb. 83

Die Funktionstabelle dieser Darstellung ist in Abb. 84 wiedergegeben:

x4	x2	y	ü
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Abb. 84

Das Addierwerk (Abb. 79) hat aber genau genommen drei Eingangsleitungen: Lx1, Lx4 und Lüv. Die Leitung Lüv führt immer den verzögerten Übertragungsimpuls, der sich bei dem Additionsvorgang in der vorausgehenden Wertstufe ergeben kann.
Um die Tabellen nach Abb. 83 und Abb. 84 zu vervollständigen, muß also auch der verzögerte Übertragsimpuls als Eingangsgröße mit berücksichtigt werden. So ergibt sich die vollständige Funktionsübersicht (Abb. 85, Abb. 86) des Addierwerkes:

Zustand auf der Leitung Lx4	Zustand auf der Leitung Lx2	Zustand auf der Leitung Lüv	Zustand auf der Leitung Ly	Zustand auf der Lü
kein Impuls	kein Impuls	kein Impuls	kein Impuls	kein Impuls
kein Impuls	kein Impuls	Impuls	Impuls	kein Impuls
kein Impuls	Impuls	kein Impuls	Impuls	kein Impuls
kein Impuls	Impuls	Impuls	kein Impuls	Impuls
Impuls	kein Impuls	kein Impuls	Impuls	kein Impuls
Impuls	kein Impuls	Impuls	kein Impuls	Impuls
Impuls	Impuls	kein Impuls	kein Impuls	Impuls
Impuls	Impuls	Impuls	Impuls	Impuls

Abb. 85

x4	x2	üv	y	ü
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Abb. 86

In den obenstehenden Tabellen (Abb. 85 bzw. Abb. 86) sind sämtliche Möglichkeiten erfaßt, die bei der Addition zweier Dualwerte zuzüglich eines Übertrages auftreten können. Jetzt ist es einfach, die Vorgänge bei der Addition des Inhaltes des Registers 4 zu dem des Registers 2 während der einzelnen Taktzeiten anzugeben (Abb. 87 bzw. Abb. 88):

Inhalt des Registers 4: ... 0 0 1 1 1 0 1 1
Inhalt des Registers 2: ... 1 0 0 1 1 0 1 0

T_8 T_7 T_6 T_5 T_4 T_3 T_2 T_1 Taktzeiten, in denen die Addition stellenweise ausgeführt wird.

Taktzeit	Zustand a. d. Leitung Lx4	Zustand a. d. Leitung Lx2	Zustand a. d. Leitung Lüv	Zustand a. d. Leitung Ly	Zustand a. d. Leitung Lü
T1	Impuls	kein Impuls	kein Impuls	Impuls	kein Impuls
T2	Impuls	Impuls	kein Impuls	kein Impuls	Impuls
T3	kein Impuls	kein Impuls	Impuls	Impuls	kein Impuls
T4	Impuls	Impuls	kein Impuls	kein Impuls	Impuls
T5	Impuls	Impuls	Impuls	Impuls	Impuls
T6	Impuls	kein Impuls	Impuls	kein Impuls	Impuls
T7	kein Impuls	kein Impuls	Impuls	Impuls	kein Impuls
T8	kein Impuls	Impuls	kein Impuls	Impuls	kein Impuls

Abb. 87

Taktzeit	x4	x2	üv	y	ü
T1	1	0	0	1	0
T2	1	1	0	0	1
T3	0	0	1	1	0
T4	1	1	0	0	1
T5	1	1	1	1	1
T6	1	0	1	0	1
T7	0	0	1	1	0
T8	0	1	0	1	0

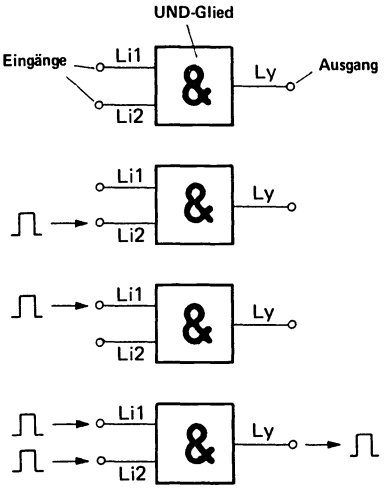
Abb. 88

Die Ergebnisbildung im Register 2 ist eng an diese Taktfolge gebunden. Einzelheiten darüber gehen aus Abb. 80 hervor. Die Leitung Ly wird nacheinander in der Taktfolge T1, T2, T3, T5. . . . von Speicherzelle zu Speicherzelle des Registers 2 fortgeschaltet. Führt sie einen elektrischen Impuls, so erfolgt ein Einschalten, andernfalls ein Ausschalten der angesteuerten Speicherzelle. Der alte Inhalt der Speicherzelle geht dabei verloren, er wird überschrieben.
Nach vollzogenen acht Takten weist das Register 2 einen Inhalt von . . . 1101 0101 auf, der mit dem Additionsergebnis (bzw. den Angaben in Spalte 5 von Abb. 88) identisch ist. Die Additionsvorgänge spielen sich in Millionstel Sekunden ab.
Nach diesen Ausführungen sollte noch geklärt werden, wie die elektronischen Schaltkreise im Inneren des Addierwerkes arbeiten. Daß dabei Transistoren, Dioden, Widerstände, Kondensatoren etc. eine große Rolle spielen, ist sicherlich schon bekannt. Diese Bauelemente werden zum Aufbau sog. UND- und ODER-Glieder (Abb. 89) verwendet, die die Grundbausteine der Computerelektronik bilden. Die Funktion der UND- und ODER-Glieder läßt sich mit wenigen Worten umschreiben:
Ein UND-Glied liefert nur dann ein Ausgangssignal (Impuls), wenn alle seine Eingangsleitungen ebenfalls ein Signal führen (Abb. 89).
Beim ODER-Glied liegen die Verhältnisse etwas anders; es sieht bereits dann schon ein Ausgangssignal vor, wenn mindestens eine seiner Eingangsleitungen ein Signal führt (Abb. 89).
Diese beiden Merksätze sind von grundlegender Bedeutung!!!
In Abb. 89 sind die verschiedenen Möglichkeiten, die beim Betrieb von UND- und ODER-Gliedern auftreten können, graphisch dargestellt und tabellarisch zusammengefaßt. Für ein UND-Glied hat sich die Kurzbezeichnung &, für ein ODER-Glied die Kurzbezeichnung v eingebürgert.
Das Addierwerk besteht aus einer Vielzahl solcher UND- und ODER-Glieder, welche nach ganz bestimmten Regeln aus den Angaben der Funktionstabelle (Abb. 86) des Addierwerkes abgeleitet und untereinander verbunden werden. Diese Regeln lassen sich am besten zunächst anhand eines kleineren Beispieles erläutern: Es besteht die Aufgabe, eine Schaltung zu entwickeln, die den Angaben der folgenden Funktionstabelle (Abb. 90) genügt:
Die beiden Eingangsgrößen sind i1 und i2. Sie können entweder den Wert „0" (kein Impuls) oder „1" (Impuls) annehmen. Die Ausgangsgröße ist mit y bezeichnet – sie hat in Abhängigkeit von den Eingangsbedingungen ebenfalls den Wert „0" (kein Impuls) oder „1" (Impuls). Nach der Methode der sogenannten disjunktiven Normalform wird aus jeder Zeile der Funktionstabelle mit einer 1 für die Ausgangsgröße (diese Zeilen sind durch * bzw. ** gekennzeichnet) ein UND-Glied abgeleitet. Ein solches UND-Glied hat für jede Eingangsgröße eine Eingangsleitung. Das UND-Glied &1 (Abb. 91) für die mit einem * markierten Zeile der Funktionstabelle (Abb. 90) läßt sich in der Formelsprache der Schaltalgebra wie folgt darstellen:

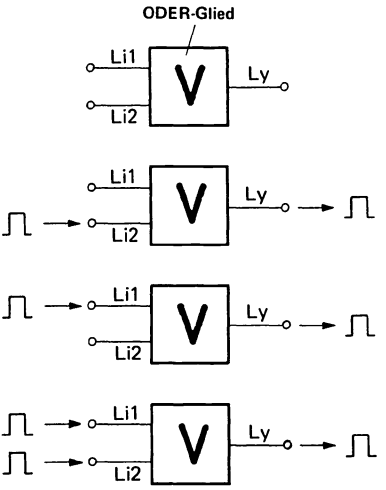
$$(\overline{i1} \,\&\, i2)$$

(Für i1 = 0 wurde $\overline{i1}$, für i2 = 1 wurde i2 gesetzt).
In diesem Ausdruck deutet das Zeichen & ein UND-Glied für die beiden Eingangsgrößen $\overline{i1}$ (in Worten: nicht i1) und i2 an.
Es ist bereits bekannt, daß ein UND-Glied nur dann ein Ausgangssignal vorsieht, wenn alle seine Eingangsleitungen einen Impuls führen.
Für i1 = 0 (kein Impuls) und i2 = 1 (Impuls) könnte sich also nie ein Ausgangssignal

UND-Glied (mit 2 Eingängen)



ODER-Glied (mit 2 Eingängen)



Zustand auf der Leitung Li1	Zustand auf der Leitung Li2	Zustand auf der Leitung Ly
Kein Impuls	Kein Impuls	Kein Impuls
Kein Impuls	Impuls	Kein Impuls
Impuls	Kein Impuls	Kein Impuls
Impuls	Impuls	Impuls

i1	i2	y
0	0	0
0	1	0
1	0	0
1	1	1

Funktionstabelle eines UND-Gliedes mit 2 Eingangsgrößen

Zustand auf der Leitung Li1	Zustand auf der Leitung Li2	Zustand auf der Leitung Ly
Kein Impuls	Kein Impuls	Kein Impuls
Kein Impuls	Impuls	Impuls
Impuls	Kein Impuls	Impuls
Impuls	Impuls	Impuls

i1	i2	y
0	0	0
0	1	1
1	0	1
1	1	1

Funktionstabelle eines ODER-Gliedes mit 2 Eingangsgrößen

Abb. 89

i1	i2	y	
0	0	0	
0	1	1	*
1	0	1	**
1	1	0	

Abb. 90

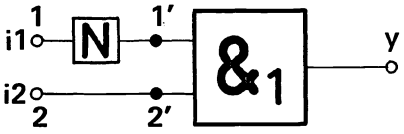
ergeben, wohl aber für den negierten Zustand von i1 = 0, d. h. $\overline{i1}$ = 1, und i2 = 1.
Die Negation einer Größe läßt sich durch einen hochgesetzten Querstrich ausdrücken. Durch Negation wird aus einer Null (kein Impuls) eine Eins (Impuls) und umgekehrt.
Die Rezeptvorschrift der disjunktiven Normalform besagt, daß für eine Eingangsgröße mit dem Wert 0 stets die Negation dieser Eingangsgröße, für eine Eingangsgröße mit dem Wert 1 stets die Eingangsgröße selbst anzugeben ist.
Auf diese Art und Weise läßt sich für die durch den * gekennzeichnete Zeile in Abb. 90 das UND-Glied &1

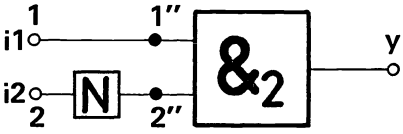
$(\overline{i1}$ & i2)

und andererseits für die durch ** gekennzeichnete Zeile das UND-Glied &2 entwickeln.

(i1 & $\overline{i2}$)

Die Erzeugung einer negierten Größe ($\overline{i1}$ od. $\overline{i2}$) stellt schaltungsmäßig kein Problem dar. Dazu werden Negationsschaltungen (N) verwendet, die entweder aus einem vorhandenen einen fehlenden Impuls machen oder umgekehrt. In der graphischen Darstellung hätten beide UND-Glieder folgendes Aussehen (Abb. 91):





($\overline{i1}$ & i2) (i1 & $\overline{i2}$)

Abb. 91

Nach der Methode der disjunktiven Normalform müssen die Ausgangsleitungen aller aus einer Funktionstabelle abgeleiteten UND-Glieder einem ODER-Glied zugeführt werden. In der Formelsprache der Schaltalgebra ausgedrückt hieße dies in unserem Beispiel:

$(\overline{i1}$ & i2) v (i1 & $\overline{i2}$)

Die entsprechende graphische Darstellung dazu sieht folgendermaßen aus: (Abb. 92)

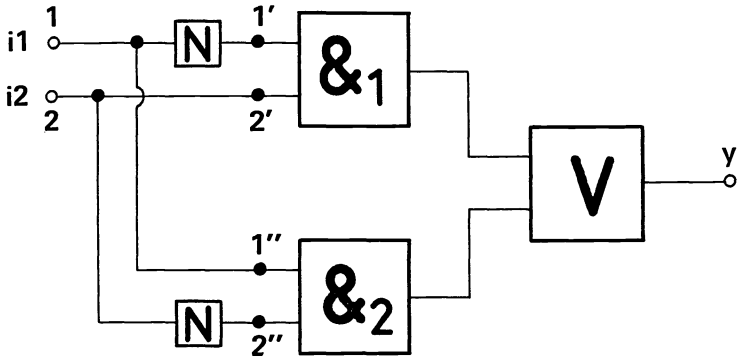


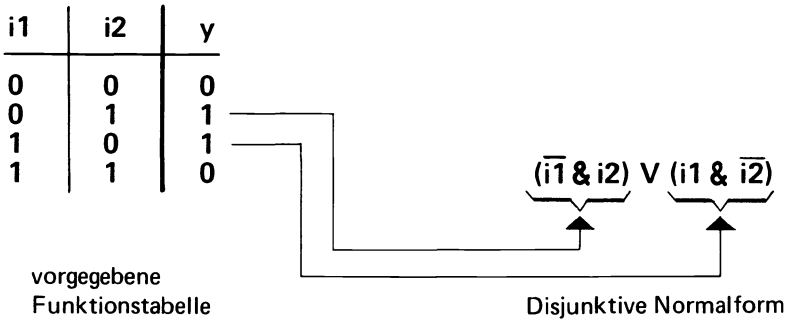
Abb. 92

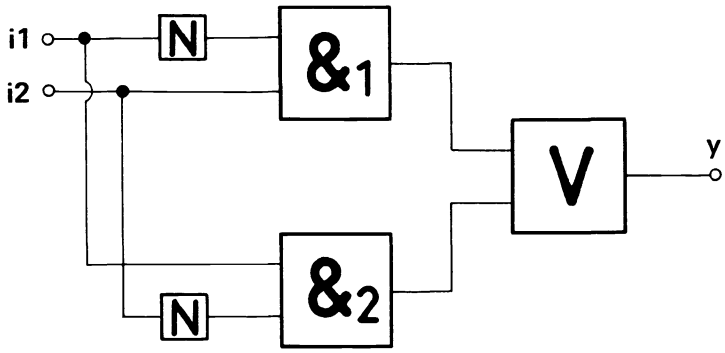
Diese Schaltung befriedigt voll und ganz die in der Funktionstabelle (Abb. 90) vorgegebenen Bedingungen. Diese Behauptung läßt sich schnell auf ihre Richtigkeit überprüfen, indem man die Probe aufs Exempel macht (Abb. 93):
Beginnen wir mit der ersten Zeile der Funktionstabelle (Abb. 90). Für i1 = 0 (kein Impuls an Klemme 1) und i2 = 0 (kein Impuls an Klemme 2) darf die Schaltung (Abb. 92), also der Ausgang des ODER-Gliedes kein Ausgangssignal (y=0) abgeben. Und fürwahr, diese Forderung trifft auch zu. Denn keines der beiden UND-Glieder erzeugt einen Ausgangsimpuls. So bleibt das ODER-Glied ohne jedes Eingagssignal, was aber die Voraussetzung für die Abgabe eines Ausgangsimpulses wäre.
Nehmen allerdings die Eingangsgrößen an den Klemmen 1 und 2 die Werte i1 = 0 und i2 = 1 (zweite Zeile der Funktionstabelle in Abb. 90) an, so gibt das UND-Glied &1 und somit auch das ODER-Glied ein Ausgangssignal ab. Wie es dazu kommt, ist leicht einzusehen. i1 = 0 wird zu $\overline{i1}$ = 1 negiert, während i2 = 1 unverändert dem UND-Glied &1 zugeführt wird. Da aber jetzt dessen Eingangswerte beide 1 sind ($\overline{i1}$ = 1 und i2 = 1) kommt es zur Erzeugung eines Ausgangsimpulses *). Das UND-Glied &2 hingegen kann keinen Ausgangsimpuls liefern, da seine Eingangsgrößen die Werte i1 = 0 und $\overline{i2}$ = 0 (= Negation von i2 = 1) haben. Das ODER-Glied erhält also nur vom UND-Glied &1 einen Impuls, was aber für die Abgabe eines Ausgangsimpulses (y = 1) bereits ausreicht *).
Nach den gleichen Überlegungen lassen sich die weiteren Zeilen der Funktionstabelle (Abb. 90) einer Betrachtung unterziehen. Dabei wird man erfreut feststellen, daß die Schaltung (Abb. 92) in allen vier Fällen genauso arbeitet (Abb. 93), wie es in der Funktionstabelle gefordert wurde. Der Weg zur Entwicklung dieser Schaltung ist zur besseren Übersicht nochmals in Abb. 94 angegeben.

* Siehe Merksätze auf Seite 106

i1 (Zustand an Klemme 1)	i2 (Zustand an Klemme 2)	Zustand am Punkt 1'	Zustand am Punkt 2'	Erzeugt das UND-Glied &1 einen Ausgangsimpuls?	Zustand am Punkt 1''	Zustand am Punkt 2''	Erzeugt das UND-Glied &2 einen Ausgangsimpuls?	Zustand am y-Ausgang des ODER-Gliedes
Kein Impuls	Kein Impuls	Impuls	Kein Impuls	nein	Kein Impuls	Impuls	nein	Kein Impuls
Kein Impuls	Impuls	Impuls	Impuls	ja	Kein Impuls	Kein Impuls	nein	Impuls
Impuls	Kein Impuls	Kein Impuls	Kein Impuls	nein	Impuls	Impuls	ja	Impuls
Impuls	Impuls	Kein Impuls	Impuls	nein	Impuls	Kein Impuls	nein	Kein Impuls

Abb. 93





Schaltung

Abb. 94

Nach der gleichen Methode der disjunktiven Normalform läßt sich auch das Addierwerk für Dualzahlen entwickeln. Die entsprechende Funktionstabelle wurde bereits aufgestellt. Sie umfaßt drei Eingangsgrößen und zwei Ausgangsgrößen (Abb. 86). Bei der Schaltungsentwicklung bezieht man sich zunächst separat auf die eine Ausgangsgröße und später auf die andere (Abb. 95); (Abb. 96).

Anschließend können beide Teilschaltungen an den gleichartigen Eingangsklemmen zusammengeführt werden (Abb. 97): Ein vollständiges Dualzahlenaddierwerk ist entstanden!

x4	x2	üv	y
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

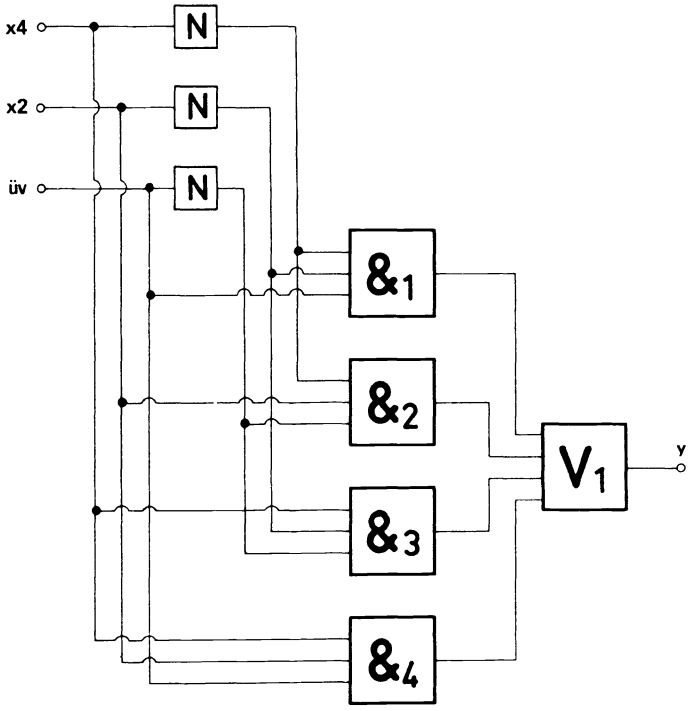
Funktionstabelle

Verknüpfung aller UND-Glieder über das ODER-Glied V1

$(\overline{x4} \,\&\, \overline{x2} \,\&\, \overline{üv}) \vee (\overline{x4} \,\&\, x2 \,\&\, \overline{üv}) \vee (x4 \,\&\, \overline{x2} \,\&\, \overline{üv}) \vee (x4 \,\&\, x2 \,\&\, üv)$

UND-Glied &1 UND-Glied &2 UND-Glied &3 UND-Glied &4

Disjunktive Normalform



Blockschaltbild eines Teiles des Dualzahlenaddierwerkes zur Bildung des Ergebnisses y ohne Übertrag

Abb. 95

x4	x2	üv	ü
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

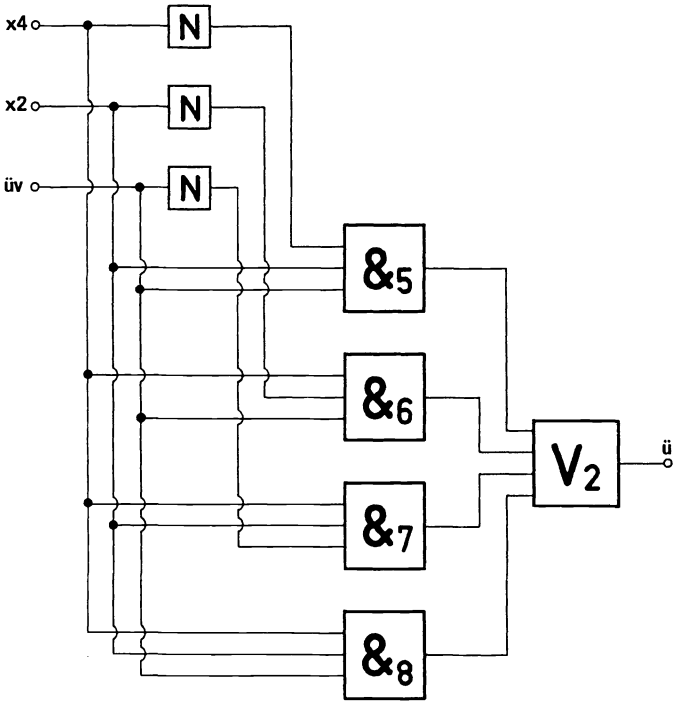
Funktionstabelle

Verknüpfung aller UND-Glieder über das ODER-Glied V2

$(\overline{x4} \,\&\, x2 \,\&\, üv) \vee (x4 \,\&\, \overline{x2} \,\&\, üv) \vee (x4 \,\&\, x2 \,\&\, \overline{üv}) \vee (x4 \,\&\, x2 \,\&\, üv)$

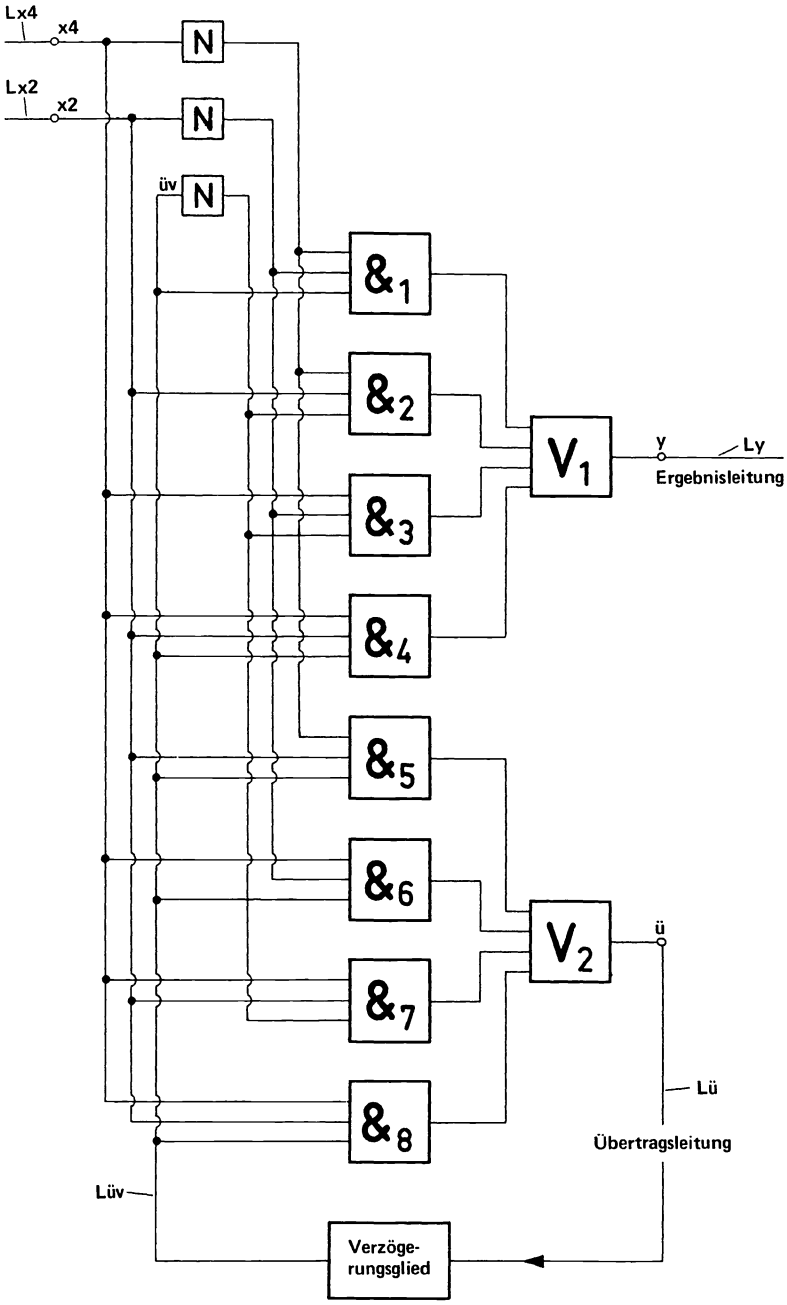
UND-Glied &5 UND-Glied &6 UND-Glied &7 UND-Glied &8

Disjunktive Normalform



Blockschaltbild eines Teiles des Dualzahlenaddierwerkes zur Bildung des Übertrages ü

Abb. 96



Blockschaltbild eines Dualzahlenaddierwerkes

Abb. 97

Selbstverständlich kann der Elektronik-Fachmann diese Schaltung noch bedeutend vereinfachen; jedoch ist dieser Umstand für die hier gemachten prinzipiellen Betrachtungen ohne große Bedeutung. Außerdem ist es auch möglich, eine solche Schaltung nach anderen Methoden als gerade über die disjunktive Normalform zu entwickeln. Aber derartige Lösungsmethoden sollen der Spezialliteratur vorbehalten bleiben.
Was allerdings noch allgemein interessant sein könnte, wäre der detaillierte Aufbau eines UND- bzw. ODER-Gliedes (Abb. 100, Abb. 101).
Zuvor sind aber noch einige Bemerkungen (Abb. 98) zum Transistor erforderlich: Dieses elektronische Bauelement hat drei Anschlüsse: den Emitter, den Kollektor und die Basis. Der Transistor in der Schaltung nach Fig. 98 wird wie ein automatischer Stromschalter betrieben. Entweder bietet er dem Strom freien Durchgang (vom Emitter zum Kollektor) oder er stellt für ihn, wie ein AUS-Schalter, ein unüberwindliches Hindernis dar. Der Transistor kann sich demnach in einem leitenden oder nicht leitenden (gesperrten) Zustand befinden. Welcher dieser beiden Zustände auftritt, hängt von der Basisvorspannung ab. Ist diese genügend positiv, ist der Transistor stromleitend, andernfalls jedoch gesperrt. Man kann also durch die Wahl der Basisvorspannung den Transistor leitend machen oder sperren.
Am besten lassen sich diese Verhältnisse mit einem Wasserhahn vergleichen. Im aufgedrehten Zustand kann ungehindert das Wasser passieren; im geschlossenen Zustand bleibt jedoch selbst dem kleinsten Tröpfchen der Weg versperrt. Die beiden Endstellungen des Drehgliedes entsprechen der Basisvorspannung für den leitenden bzw. gesperrten Transistor, das Wasser entspricht dem elektrischen Strom.
Im leitenden Zustand fließt der Strom von Masse über den Transistor (vom Emitter zum Kollektor) und dessen Arbeitswiderstand Ra zum positiven Pol einer Spannungsquelle (Batterie) – (Abb. 98). Dabei tritt am Arbeitswiderstand Ra ein Spannungsabfall auf. Während im nichtleitenden Fall des Transistors die Kollektorspannung der positiven Batteriespannung + Ub entspricht, nimmt jetzt, durch den auftretenden Spannungsabfall

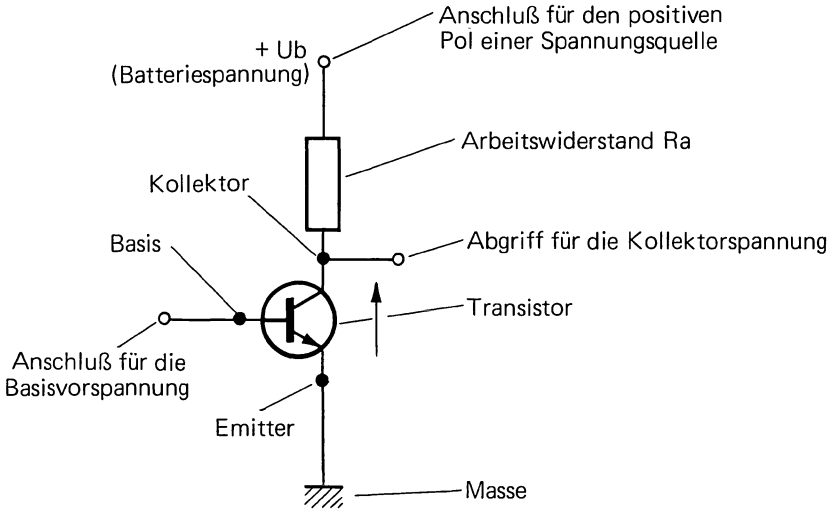


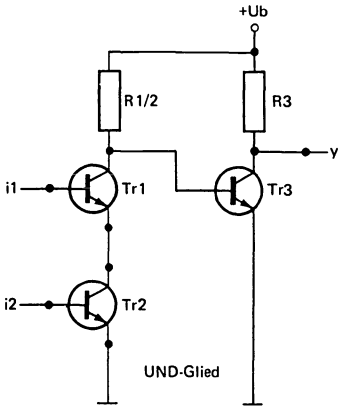
Abb. 98

bedingt, die Kollektorspannung einen bedeutend niedrigeren Wert an. Diese Ausführungen sind in der tabellarischen Aufstellung nach Abb. 99 zusammengefaßt.

Zustand des Transistors	Basisvorspannung	Kollektorspannung
leitend	(ausreichend) positiv	<+ Ub (bedeutend kleiner als die volle Batteriespannung)
gesperrt	(nicht ausreichend positiv oder) negativ	+ Ub (volle Batteriespannung)

Abb. 99

Beim Übergang des Transistors aus dem gesperrten in den leitenden Zustand durch Anlegen einer genügend positiven Basisvorspannung tritt am Kollektor ein negativer Spannungssprung (von der vollen Batteriespannung +Ub zu einem bedeutend kleineren Wert) auf.
Beim Übergang des Transistors von dem leitenden in den gesperrten Zustand durch Fortfall der ausreichend positiven Basisvorspannung erfährt der Kollektor hingegen einen positiven Spannungssprung.
Diese Kenntnisse reichen bereits aus, um sich die Wirkungweise von UND- und ODER-Gliedern zu erklären. Es sei vorangestellt, daß die Transistoren in diesen Schaltungen durch kurzzeitige elektrische (Basis-)Impulse in den leitenden oder gesperrten Zustand gesteuert werden.
Dabei soll die Zuordnung gelten, daß ein positiver Impuls als binäre „1", ein fehlender positiver Impuls als binäre „0" aufzufassen ist.
Das UND-Glied nach Abb. 100 ist für zwei Eingangsgrößen i1 und i2 ausgelegt, die als vorhandene oder fehlende positive Impulse auftreten können. Durch einen positiven Impuls an der Basis gelangt der betreffende Transistor kurzzeitig (für die Dauer des Impulses) in den leitenden Zustand.
Angenommen, die Transistoren Tr 1 und Tr 2 wären gesperrt (i1 = 0 = kein Impuls; i2 = 0 = kein Impuls). Der Widerstand R1/2 bliebe deshalb stromfrei und die Kollektorspannung des Transistors Tr 1 hätte dann den Wert der vollen Batteriespannung +Ub. Wegen der Verbindung des Kollektors des Transisotrs Tr 1 mit der Basis des Transistors TR 3 (Abb. 100) ist die Basisvorspannung des Transistors TR 3 immer gleich der Kollektorspannung des Transistors Tr 1, in diesem Falle also +Ub. Dieser Wert ist jedoch genügend positiv, um den Transistor Tr 3 im leitenden Zustand zu halten. Folglich ist der Widerstand R3 stromdurchflossen und die Kollektorspannung des Transistors Tr 3 bedeutend geringer als die volle Batteriespannung +Ub.
An diesen Verhältnissen ändert sich auch nichts, wenn an die Basis nur eines der beiden Transistoren Tr 1 bzw. Tr 2 ein positiver Impuls gelangt. Das wäre z.B. für i1 = 1, i2 = 0 oder i1 = 1, i2 = 0 der Fall. Der durch den Impuls angesteuerte Transistor ist zwar bereit, Strom hindurchzulassen. Doch der andere Transistor bleibt weiterhin gesperrt. Und deshalb kann auch kein Strom von Masse über die beiden Transistoren Tr 2 und Tr 1 und den Widerstand R1/2 zum positiven Pol der Spannungsquelle (Batterie) fließen.
Ein solcher Stromfluß ist nur dann möglich, wenn beide Transistoren Tr 1 und Tr 2 gleichzeitig durch jeweils einen positiven Basisimpuls angesteuert werden (i1 = 1 = Impuls; i2 = 1 = Impuls). Denn dann befänden sich beide Transistoren gleichzeitig im leitenden Zustand und der Strom könnte sie ungehindert passieren. Das würde aber ein Absinken der Kollektorspannung auf einen bedeutend kleineren Wert als die volle



Zuordnung:

positiver Impuls = "1"
kein positiver Impuls = "0"

i1	i2	y
0	0	0
0	1	0
1	0	0
1	1	1

Funktionstabelle

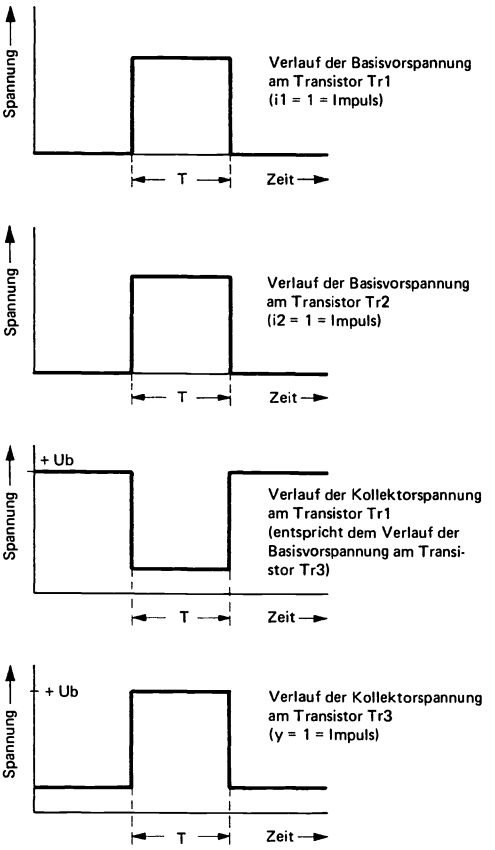


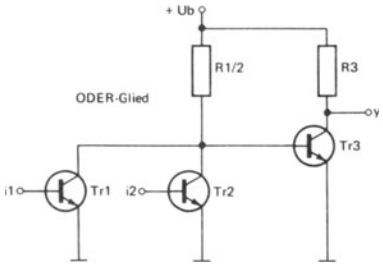
Diagramm der Spannungsverläufe des UND-Gliedes für den Fall i1 = i2 = y = 1

Abb. 100

Batteriespannung +Ub bedeuten. Dieser Wert wäre jedoch nicht mehr genügend groß genug, um den Transistor Tr 3 weiterhin in seinem leitenden Zustand zu halten. Folglich geht der Transistor Tr 3 während der Impulszeit T in den gesperrten Zustand. Der Widerstand R3 bleibt während dieser Zeit stromlos und die Kollektorspannung des Transistors Tr 3 nimmt vorübergehend den vollen Wert der Batteriespannung an. Diese kurzzeitige Spannungserhöhung entspricht aber einem positiven Impuls (y = 1).
Diese Vorgänge sind in Abb. 100 zum besseren Verständnis auch noch graphisch illustriert.
Nach diesen Betrachtungen wird es um so leichter fallen, die Funktion eines ODER-Gliedes (Abb. 101) zu verstehen. Im Ruhestatus des ODER-Gliedes (i1 = 0 = kein Impuls; i1 = 0 = kein Impuls) sind die Transistoren Tr 1 und TR 2 im gesperrten

Zustand. Weder der eine noch der andere von ihnen liefert Strom an den Widerstand R1/2. Folglich hat die Spannung an den miteinander verbundenen Kollektoren dieser beiden Transistoren den Wert der vollen Batteriespannung +Ub. Wegen der Verbindung des Kollektors der Transistoren Tr 1 und Tr 2 mit der Basis des Transistors Tr 3 ist die Basisvorspannung des Transistors 3 immer gleich der Kollektorspannung der Transistoren Tr 1 und Tr 2, in diesem Falle also +Ub. Dieser Wert +Ub ist genügend positiv, um den Transistor Tr 3 in seinem leitenden Zustand zu halten. Folglich ist der Widerstand R 3 stromdurchflossen und die Kollektorspannung des Transistors Tr 3 bedeutend geringer als die volle Batteriespannung +Ub.
Diese Verhältnisse ändern sich aber genau in dem Moment, wenn an die Basis mindestens eines der beiden Transistoren Tr 1 oder Tr 2 ein positiver Impuls gelangt. Ein solcher Impuls steuert den betreffenden Transistor in den leitenden Zustand und bewirkt einen Stromfluß durch den Widerstand R1/2. Durch den an R1/2 auftretenden Spannungsabfall sinkt die Kollektorspannung dieses Transistors auf einen bedeutend kleineren Wert als die volle Batteriespannung. Wegen der Verbindung ihrer Kollektoren müssen beide Transistoren zwangsweise die gleiche Kollektorspannung zeigen, denn an einem Punkt kann nur eine einzige Spannung auftreten. Dieser Hinweis erscheint nötig, denn sonst käme man leicht in gedankliche Schwierigkeiten. So könnte zum Beispiel der leitende Transistor Tr 1 für sich betrachtet eine kleinere Kollektorspannung bedingen als der nichtleitende Transistor Tr 2. Die Frage, welche dieser Spannungen nun den Vorrang hat, ist mit drei Worten beantwortet: Immer die kleinere. Sobald also mindestens einer der Transistoren Tr 1 oder Tr 2 stromleitend ist, wird an deren gemeinsamen Kollektoren immer eine Spannung auftreten, die einen bedeutend kleineren Wert als die volle Batteriespannung +Ub hat.
Diese Kollektorspannung ist aber mit der Basisvorspannung des Transistors Tr 3 identisch.
Im Ruhestatus des ODER-Gliedes war der Transistor Tr 3 stromleitend. Mit dem Auftreten von Eingangsimpulsen (i1 = 1 oder i2 = 1) sinkt jedoch die Kollektorspannung der Transistoren Tr 1 und Tr 2 und damit auch die Basisvorspannung des Transistors Tr 3 unter den für dessen leitenden Zustand erforderlichen Wert. Folglich wird der Transistor Tr 3 während der Impulszeit T gesperrt; seine Kollektorspannung nimmt vorübergehend den vollen Wert der Batteriespannung an. Diese kurzzeitige Spannungserhöhung entspricht aber dem erhofften positiven Impuls. Schließlich sollte ja das ODER-Glied immer dann einen Ausgangsimpuls liefern, wenn mindestens an einem seiner Eingänge ein Impuls auftritt. Diese Ausführungen sind durch eine entsprechende graphische Darstellung in Abb. 101 untermauert.
Auf den vorstehenden Seiten wurde die Computer-Elektronik aus verständlichen Gründen ziemlich populär dargestellt; doch nur so konnte der Nichtfachmann erfahren, was sich nun wirklich in einem Computer abspielt. Es gehört zwar nicht zu den Aufgaben dieses Buches, auf elektronische Schaltungen einzugehen, aber vielleicht war es doch einmal interessant, diesen kleinen Ausflug in das „Draht- und Transistorenverhau" des Computers zu unternehmen. Selbstverständlich kennt der Fachmann bedeutend mehr elektronische Bauelemente und Methoden zur Entwicklung von Schaltkreisen, als hier angedeutet wurde. In diesem Kapitel ging es lediglich darum, an einem einfachen Beispiel aufzuzeigen, mit welchen primitiven Informationen (vorhandene oder fehlende Impulse) ein Computer arbeiten muß und welche Mittel dafür zur Verfügung stehen. Und wenn es jetzt darum ginge, den detaillierten Schaltplan für das Dualzahlen-Addierwerk (Abb. 97) anzugeben, einschließlich aller Transistoren und Widerstände, so wäre diese Aufgabe mit einiger Fleißarbeit ohne Verständnisschwierigkeiten schnell gelöst.

Die betreffenden UND- und ODER-Glieder müßten nur für mehrere Eingangsvariable (x4, x2, üv) ausgelegt und entsprechend dem Blockschaltbild nach Abb. 97 untereinander verbunden werden. Dieser Schaltung würde man natürlich nicht ansehen, daß sie in einer Sekunde mehrere Millionen Additionen ausführen kann. Aber es ist wirklich so — und darin liegt auch der große Vorteil des des Computers, der zwar primitiv und einfach rechnet, dafür aber so schnell, daß er jeden Kopfrechner weit hinter sich lassen würde.



Zuordnung:

positiver Impuls = "1"
kein positiver Impuls = "0"

i1	i2	y
0	0	0
0	1	1
1	0	1
1	1	1

Funktionstabelle

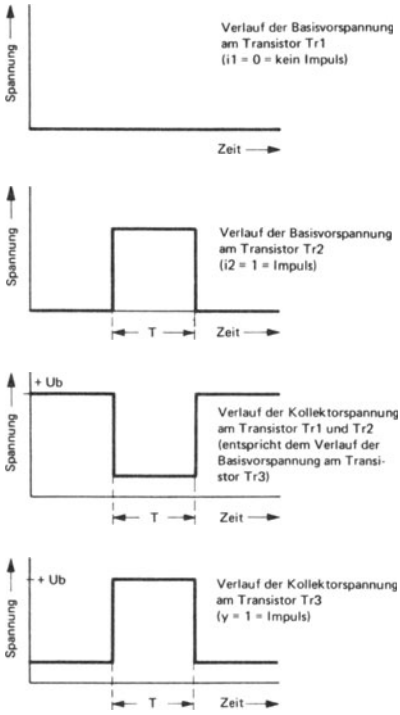


Diagramm der Spannungsverläufe des ODER-Gliedes für den Fall i1 = 0; i2 = 1; y = 1

Abb. 101

14. Ein „Gruppenwechsel", programmtechnisch betrachtet

Wünscht sich z. B. ein Kaufhausdirektor eine Umsatzstatistik für einzelne Warengruppen, so könnte sich sein Wunsch ohne weiteres mit Hilfe eines Programmes erfüllen lassen. Dazu werden alle Kaufhausartikel zu Warengruppen zusammengefaßt, z. B.:
Öle, Schmalz und Butter zur Warengruppe Fette mit der Nr. 01 und Rollmöpse, Sprotten und Kaviar zur Warengruppe 03 für Fischwaren.

Die vorhandenen Artikel-Umsatz-Lochkarten (Abb. 102) weisen folgende Feldeinteilung auf:

Spalte 1 – 4	Artikelnummer
Spalte 6 – 7	Nummer der Warengruppe
Spalte 9 – 30	Artikelbezeichnung
Spalte 31 – 35	Umsatz
Spalte 38 – 80	(in diesem Zusammenhang uninteressant)

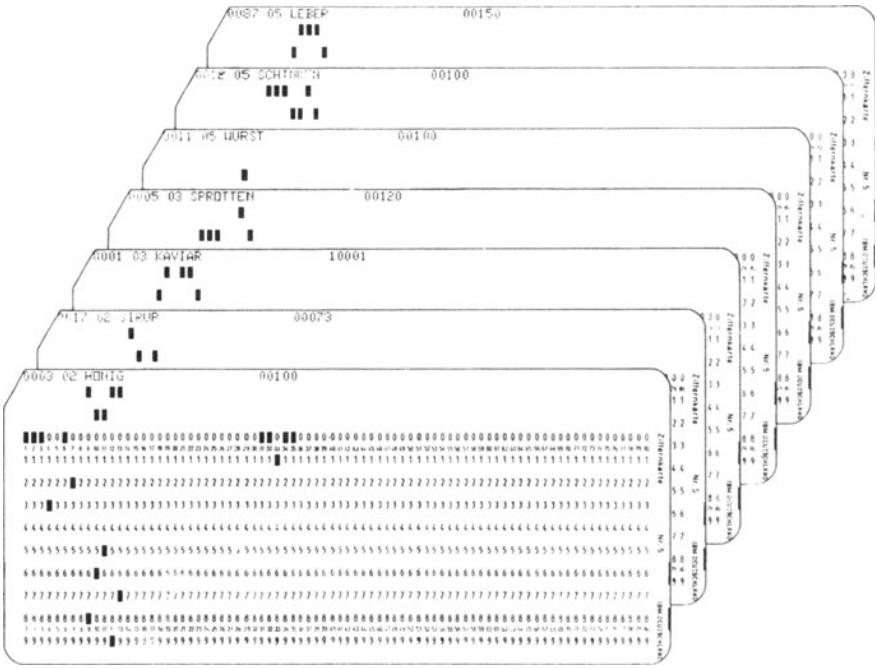


Abb. 102

In der erwünschten Liste (Abb. 103) sollen folgende Angaben erscheinen:

Druckstelle 5 – 6	Warengruppe
Druckstelle 12 – 16	Gesamtumsatz für alle Artikel dieser Warengruppe.

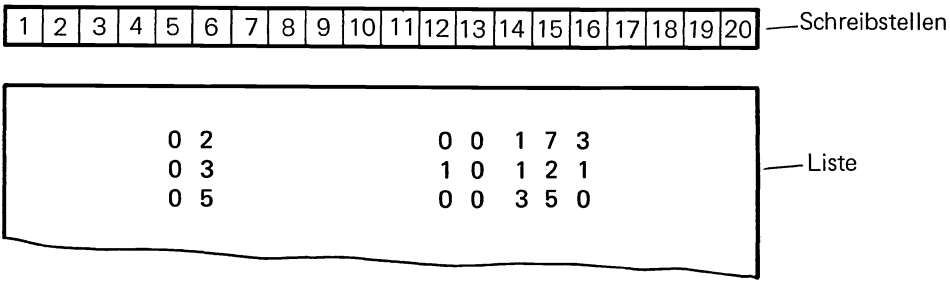


Abb. 103

Aus diesem Grunde müssen im Programm die Umsätze aller Artikel der gleichen Warengruppe addiert werden. Dieser Forderung läßt sich besonders leicht nachkommen, wenn die Artikel-Umsatz-Lochkarten nach der Warengruppe sortiert vorliegen (Abb. 102). Dann braucht man im Programm nur noch ein Rechenfeld vorzusehen, in das die Umsatzwerte nacheinander zu addieren sind. Diese fortlaufende Addition bezieht sich allerdings nur auf die gleiche Warengruppe. Sobald Lochkarten mit einer neuen Warengruppe auftreten – man spricht in diesem Falle von einem Gruppenwechsel – ist

a) die aufgelaufene Summe des Rechenfeldes als Gesamtumsatz für die vorausgegangene Warengruppe auszudrucken

b) der Inhalt des Rechenfeldes zu löschen (auf Null zurückzustellen), um mit der fortlaufenden Addition der Umsatzwerte für die neue Warengruppe beginnen zu können.

Und nun verfolgen Sie bitte anhand des Blockdiagramms in Abb. 104, wie ein solcher Gruppenwechsel vollzogen werden kann: Der Karteneinlesebereich KARBER umfaßt 80 Stellen. Er enthält die Felder WGRU für die Warengruppe und UMS für den Artikel-Umsatz (Abb. 106). Nach dem Einlesen der allerersten Lochkarte ist der Schalter SCH1 nicht gesetzt. Zu diesem Zeitpunkt hat er laut Definition noch den Inhalt „A" (und nicht etwa „G") und das Programm folgt dem gestrichelt angedeuteten Weg zur Instruktion

EINMAL Übertrage den Inhalt des Feldes WGRU in das Feld ZWGRU

Es sei schon hier bemerkt, daß diese Instruktion EINMAL während des weiteren Programmverlaufes kein zweites Mal ausgeführt werden soll. Aus diesem Grunde gibt man anschließend dem Schalter SCH1 den Inhalt „G", so daß später die Frage

Steht im Schalter SCH1 ein „G"?

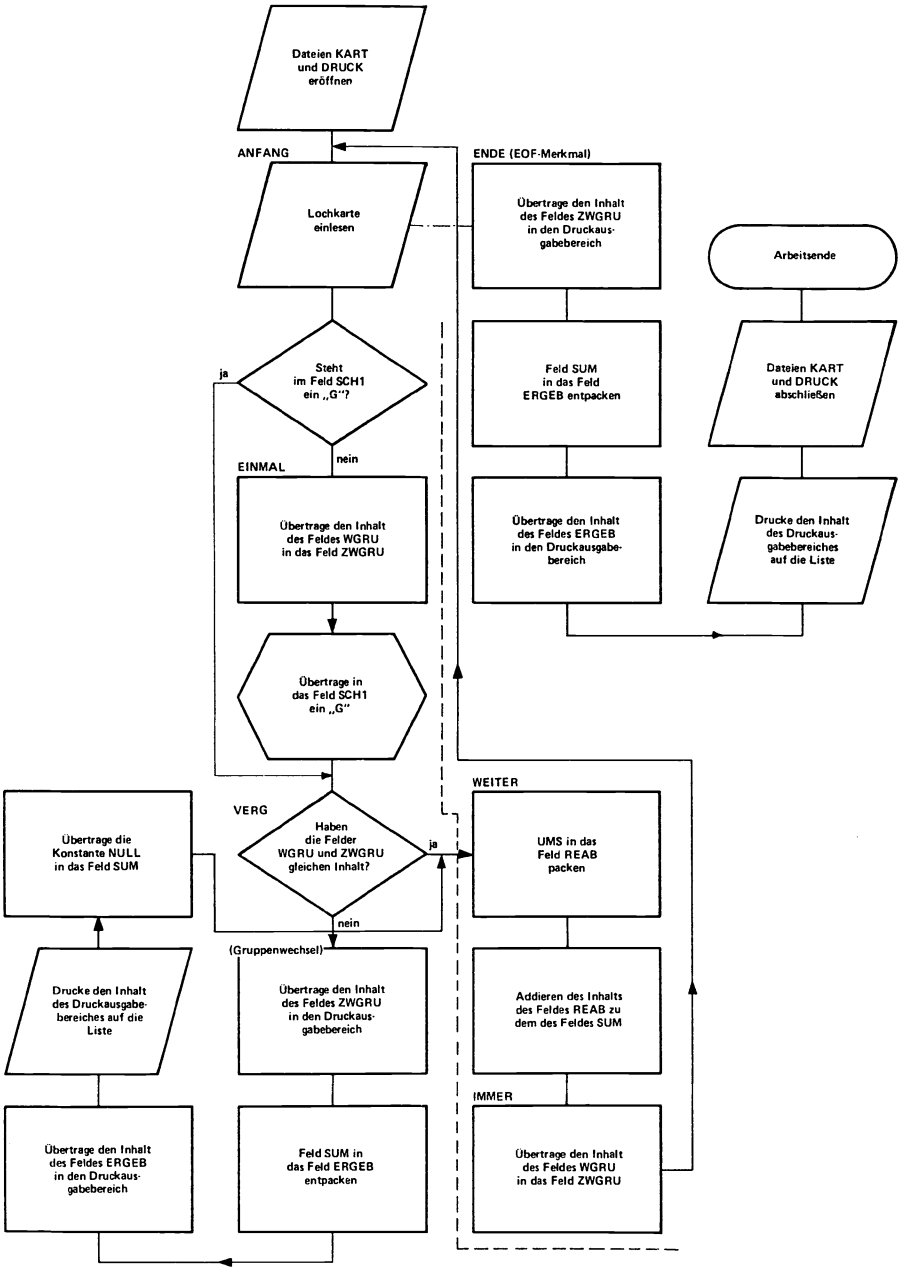

Dateien KART und DRUCK eröffnen
ANFANG
Lochkarte einlesen
Steht im Feld SCH1 ein „G"?
ja
nein
EINMAL
Übertrage den Inhalt des Feldes WGRU in das Feld ZWGRU
Übertrage in das Feld SCH1 ein „G"
VERG
Haben die Felder WGRU und ZWGRU gleichen Inhalt?
ja
nein
(Gruppenwechsel)
Übertrage die Konstante NULL in das Feld SUM
Drucke den Inhalt des Druckausgabebereiches auf die Liste
Übertrage den Inhalt des Feldes ZWGRU in den Druckausgabebereich
Übertrage den Inhalt des Feldes ERGEB in den Druckausgabebereich
Feld SUM in das Feld ERGEB entpacken
WEITER
UMS in das Feld REAB packen
Addieren des Inhalts des Feldes REAB zu dem des Feldes SUM
IMMER
Übertrage den Inhalt des Feldes WGRU in das Feld ZWGRU
ENDE (EOF-Merkmal)
Übertrage den Inhalt des Feldes ZWGRU in den Druckausgabebereich
Feld SUM in das Feld ERGEB entpacken
Übertrage den Inhalt des Feldes ERGEB in den Druckausgabebereich
Arbeitsende
Dateien KART und DRUCK abschließen
Drucke den Inhalt des Druckausgabebereiches auf die Liste

Abb. 104

stets zu bejahen ist. Der Ja-Zweig dieser Frage umgeht sodann die Instruktion EINMAL; die Begründung dafür wird später angegeben.
Die Übertragung der Warengruppennummer aus dem Feld WGRU in den Zwischenbereich ZWGRU durch die Instruktion IMMER ist von besonderer Bedeutung. Unter der Annahme, es wären bereits mehrere Lochkarten verarbeitet worden, enthält das Feld WGRU stets die Warengruppe der gerade eingelesenen Lochkarte, das Feld ZWGRU die ihrer Vorgängerin. Ein Vergleich des Inhaltes beider Felder könnte dann ergeben, daß es sich noch um die gleiche Wartengruppe handelt oder nicht (Gruppenwechsel).
Eine detaillierte Betrachtung wird diese Ausführungen verständlicher machen:
Die Warengruppe der zuletzt eingelesenen Lochkarte steht immer im Feld WGRU des Karteneinlesebereiches KARBER. Um zu verhindern, daß dieser Wert beim Einlesen der nachfolgenden Lochkarte durch Überschreiben verlorengeht, überträgt man ihn zuvor durch die Instruktion

IMMER Übertrage den Inhalt des Feldes WGRU in das Feld ZWGRU

in den Zwischenbereich ZWGRU. Nach dem Einlesen der neuen Lochkarte steht also deren Warengruppe im Feld WGRU, während im Feld ZWGRU die „alte" Warengruppe der vorausgehenden Lochkarte sichergestellt ist. Somit kann jetzt durch die Instruktion

VERG Haben die Felder WGRU und ZWGRU den gleichen Inhalt?

festgestellt werden, ob bei dem Einlesen der neuen Lochkarte ein Gruppenwechsel auftrat oder nicht.
Für die allererste Lochkarte, die ja keine Vorgängerin haben kann, soll die Frage VERG mit ja zu beantworten sein (kein Gruppenwechsel). Aus diesem Grunde ist es aber erforderlich, zuvor dem Zwischenbereich ZWGRU mit der Instruktion

EINMAL Übertrage den Inhalt des Feldes WGRU in das Feld ZWGRU

den Inhalt des Feldes WGRU zuzuweisen.
Wie bereits erwähnt, darf diese Instruktion EINMAL nur für die allererste Lochkarte ausgeführt werden. Dieser Forderung läßt sich durch Verwendung des Schalters SCH1 leicht nachkommen.
Nach dem Vergleich

VERG Haben die Felder WGRU und ZWGRU den gleichen Inhalt?

(die Felder WGRU und ZWGRU enthalten ensprechend der ersten Lochkarte in Abb. 102 beide eine „02") wird der Artikelbetrag UMS in das Feld REAB gepackt und von dort aus in das Rechenfeld SUM addiert *). Anschließend gelangt man zu der bereits erwähnten Instruktion

IMMER Übertrage den Inhalt des Feldes WGRU in das Feld ZWGRU

* Die dezimale Additionsoperation
Addieren des Inhaltes des Feldes REAB zu dem des Feldes SUM
setzt gepackte Zahlenfelder (REAB und SUM) voraus.

Für die allererste Lochkarte wäre dieser Befehl als bloße Wiederholung zur Instruktion EINMAL natürlich völlig überflüssig. Bei allen nachfolgenden Lochkarten hingegen ändert sich die Situation: Aufgrund des gesetzten Schalters SCH1 wird nämlich die Instruktion EINMAL nicht mehr ausgeführt, so daß jetzt der mit IMMER bezeichnete Befehl zum Tragen kommt.
Sollte die Frage

VERG Haben die Felder WGRU und ZWGRU den gleichen Inhalt?

eine verneinende Antwort nach sich ziehen, also einen Gruppenwechsel andeuten, so ist die „alte" Warengruppe (aus ZWGRU) mit dem zugehörigen Summenbetrag *) auf der Liste auszudrucken. Anschließend ist das Feld SUM zu löschen (auf Null zurückzustellen), damit die Addition der Umsatzwerte für die neue Warengruppe wieder von vorn (bei Null) beginnen kann.
Das Programm verzweigt danach zur Instruktion

WEITER UMS in das Feld REAB packen

und wird, wie gewohnt, für die eingelesene Lochkarte mit der neuen Warengruppe fortgesetzt.
Bei Karten-Ende ist wie bei einem Gruppenwechsel zu verfahren; nur daß anschließend noch die beteiligten Dateien abzuschließen sind.
Abb. 105 zeigt in vereinfachter Form den Inhalt der im Programm benutzten Hauptspeicherfelder zu verschiedenen Zeitpunkten (es wurde vorausgesetzt, daß die in Abb. 102 dargestellten Lochkarten gemäß dem Blockdiagramm nach Abb. 104 verarbeitet werden). Beim Nachkontrollieren der Angaben aus Abb. 105 erfährt man nochmals die Auswirkungen des programmierten Gruppenwechsels.
Unerwähnt geblieben ist bisher, wie ein größerer Hauptspeicherbereich (in diesem Fall handelt es sich um den Karteneinlesebereich KARBER) definitonsgemäß weiter unterteilt werden kann (Abb. 106).

```
KARBER    D S    0CL80
          D S    CL5
WGRU      D S    CL2
          D S    CL23
UMS       D S    CL5
          D S    CL45
```

Die gewünschte Unterteilung ist durch eine Null vor der Größenangabe CL80 kenntlich zu machen:

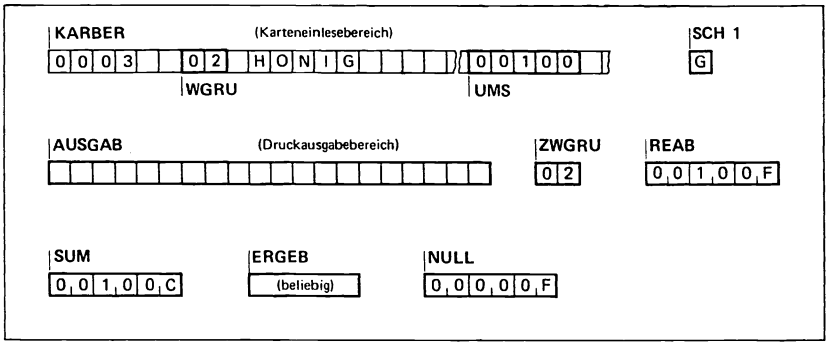
```
KARBER    D S    0CL80
```

Alle Teilbereiche müssen zusammengenommen die gleiche Länge wie der Gesamtbereich KARBER aufweisen.

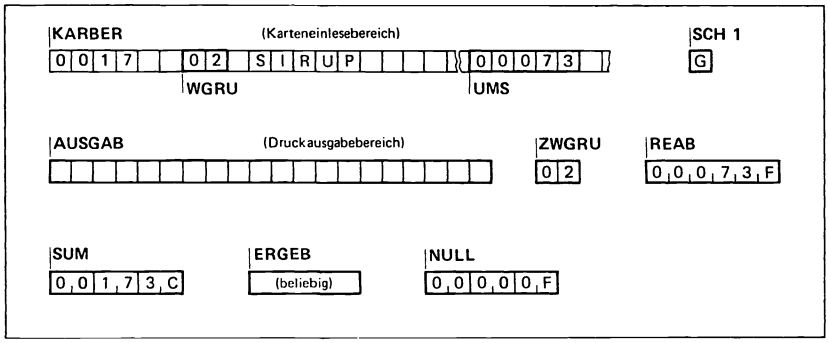
* Der Inhalt des Feldes SUM ist vor dem Ausdrucken in das Feld ERGEB zu entpacken.

Speicherauszüge zu einem Zeitpunkt

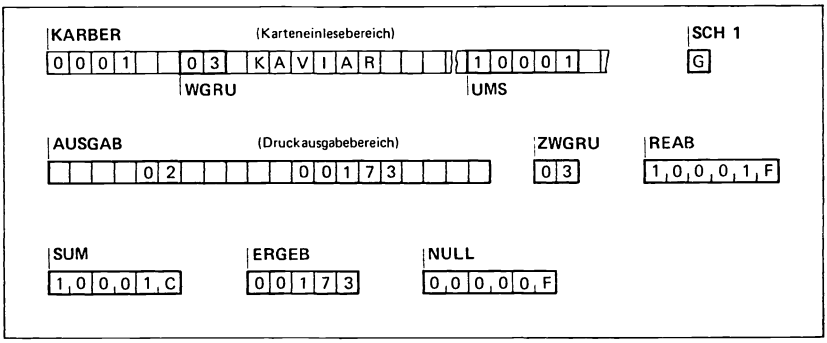
a) vor dem Einlesen der zweiten Lochkarte



b) vor dem Einlesen der dritten Lochkarte



c) vor dem Einlesen der vierten Lochkarte



(für die halbbyteweise Darstellung wurde ein größerer Maßstab verwendet)

Abb. 105

Der erste Teilbereich (Stelle 1 bis 5) braucht keinen symbolischen Namen zu erhalten, da dessen Angaben für dieses Programm uninteressant sind.

```
        DS      CL5
```

In den nachfolgenden beiden Stellen (6 bis 7) steht die Warengruppennummer, auf die man sich durch den symbolischen Namen WGRU beziehen möchte:

```
WGRU    DS      CL2
```

Den daran anschließenden 23 Stellen (8 bis 30) soll wiederum kein Name zugeordnet werden:

```
        DS      CL23
```

Das Feld für die Umsatzwerte (Stelle 31 bis 35) erhält durch die Definitionsanweisung

```
UMS     DS      CL5
```

den symbolischen Namen UMS und eine Größe von 5 Bytes zugewiesen.
Die restlichen verbleibenden (hier nicht interessierenden) Stellen 36 bis 80 des Karteneinlesebereiches sind durch den Ausdruck

```
        DS      CL 45
```

erfaßt.
Mit Hilfe dieser Kenntnisse ist bis jetzt möglich, daß symbolische Programm (Abb. 107) aufzustellen.
Durch dieses kleine Beispiel darf aber keinesfalls der Eindruck entstehen, daß ein Gruppenwechsel nur bei Warengruppen auftreten könne. Es ließen sich beliebig viele andere

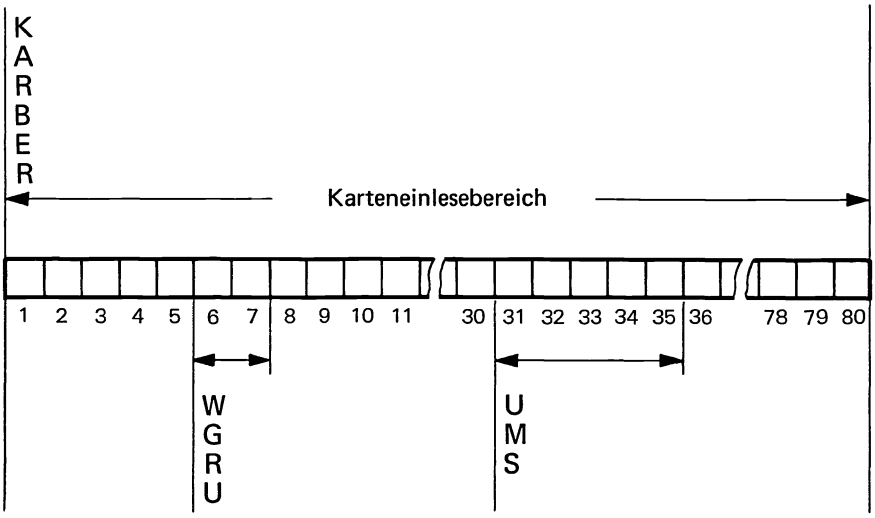


Abb. 106

SIEMENS

Assembler

Programm: GRUPPEN-WECHSEL

Bearbeiter: ☎ Datum:

Problem: Blatt 01

Name	Operation	Operanden und Bemerkungen
		Dateibeschreibungen
		Für die Lochkarten auf der Kartenabfühleinheit
		Name der Datei KART
		Name des Karten-einlesebereiches KARBER
		Art der Datei INPUT
		EOF-Merkmal ENDE
		Für die auf dem Drucker auszugebende Liste
		Name der Datei DRUCK
		Name des Druck-ausgabebereiches AUSGAB
		Art der Datei OUTPUT
	OPEN	KART,DRUCK
ANFANG	GET	KART
	CLI	SCH1,C'G'
	BE	VERG
EINMAL	MVC	ZWGRU,WGRU
	MVI	SCH1,C'G'
VERG	CLC	WGRU,ZWGRU
	BE	WEITER
	MVC	AUSGAB+4(2),ZWGRU
	UNPK	ERGEB,SUM
	MVC	AUSGAB+11(5),ERGEB
	PUT	DRUCK
	MVC	SUM,NULL
WEITER	PACK	READ,UMS
	AP	SUM,READ

Spalten 73—78 siehe Kopfzeile

Mehrfachlochungen sind in derselben Zeile als Bemerkung unter Angabe der Lochkombination zu erläutern.
Die Spalte 80 ist nur bei Einschiebungen auszufüllen. Es sollte mit „A" begonnen werden. Einschübe können in die letzten sechs Zeilen geschrieben werden.

Abb. 107a

SIEMENS

Assembler

Programm: GRUPPEN-WECHSEL

Bearbeiter: _____ Datum: _____

Problem Blatt: 02

Name	Operation	Operanden und Bemerkungen
IMMER	MVC	ZWGRU,WGRU
	B	ANFANG
ENDE	MVC	AUSGAB+4(2),ZWGRU
	UNPK	ERGEB,SUM
	MVC	AUSGAB+11(5),ERGEB
	PUT	DRUCK
	CLOSE	KART,DRUCK
	EOJ	
**DEFINITIONEN		
KARBER	DS	0CL80
	DS	CL5
WGRU	DS	CL2
	DS	CL23
UMS	DS	CL5
	DS	CL45
SCH1	DC	CL'A'
AUSGAB	DC	CL20' '
ZWGRU	DS	CL2
REAB	DS	CL3
SUM	DC	X'00000F'
ERGEB	DS	CL5
NULL	DC	X'00000F'

Spalten 73—78 siehe Kopfzeile

Mehrfachlochungen sind in derselben Zeile als Bemerkung unter Angabe der Lochkombination zu erläutern.
Die Spalte 80 ist nur bei Einschiebungen auszufüllen. Es sollte mit „A" begonnen werden. Einschübe können in die letzten sechs Zeilen geschrieben werden.

Abb. 107b

Beispiele aufführen, in denen sich ein Gruppenwechsel, je nach der vorliegenden Aufgabe entweder auf Kundennummern, Filialnummern oder andere Angaben bezieht. Eines ist allen diesen Fällen immer gemeinsam: Die fortlaufende, gruppenweise Erfassung der Daten.
Es bleibt Aufgabe des Programmes, einen Gruppenwechsel selbsttätig zu erkennen. Wie dies im einzelnen ermöglicht wird, geht aus dem vorstehenden Beispiel hervor. Ohne den Vergleich des Gruppenbegriffes zweier aufeinanderfolgender Lochkarten könnte der Gruppenwechsel nie vollzogen werden. Dazu ist aber stets die Zwischenspeicherung des vorausgehenden Gruppenbegriffes erforderlich. Daß die allererste Lochkarte in diesem Zusammenhang eine gewisse Sonderstellung einnehmen muß, versteht sich von selbst.

15. Instruktionsformate

In den vorangegangenen Kapiteln wurden bereits viele ASSEMBLER-Instuktionen behandelt. Mit wenigen Ausnahmen zeigen diese im Prinzip alle den gleichen Aufbau: Dem mnemonischen Befehlsschlüssel folgen immer zwei Operanden, die durch ein Komma voneinander getrennt sind.
Als Operanden traten bisher symbolische Namen, Registernummern oder sogar Direktwerte auf (Abb. 108).

mnemonischer Befehlsschlüssel (Operation)	Operand 1, Operand 2
MVC	AUSGABE+3(16),KARBER+1
MVC	SUM,NULL
LA	9,TAB
CLI	SCH1,C'G'
CLC	WGRU,ZWGRU
AR	2,4

Abb. 108

Jeder Operand bezeichnet zumeist einen bestimmten Platz im Kernspeicher. Man unterscheidet Register-, Speicher- und Direktoperanden. Erstere treten als Registernummern auf, während die symbolischen Feldnamen in Abb. 108 als Speicheroperanden anzusehen sind. (Den Speicheroperanden werden im echten Programm immer Adressen zugeordnet.) Direktoperanden hingegen umfassen einstellige Zeichen, die in entsprechenden (MVI-, CLI-)Instruktionen direkt angegeben werden. In vielen Fällen ist so etwas sehr vorteilhaft, denn dadurch bleibt einem die Definition eines einstelligen Hauptspeicherfeldes entsprechenden Inhaltes erspart.
Der Schreibweise der ASSEMBLER-Instruktionen liegen strenge Richtlinien zugrunde. Danach ist es zum Beispiel nicht möglich, eine MVC-Operation auf Register anzuwenden:

```
MVC     3,8        falsch!!!
```

Der Computer wäre total überfordert, sollte er etwa nach dieser Anweisung den Inhalt des Registers 8 in das Register 3 übertragen. MVC-Operationen lassen sich nun einmal nicht auf Register *) beziehen, sondern nur auf gewöhnliche Hauptspeicherplätze. Die gleiche Vorschrift gilt auch für eine Reihe anderer Instruktionen (CLC, PACK, AP, SP, MP, DP). Da in diesen Instruktionen stets nur Speicheroperanden auftreten

Operation Speicheroperand 1,Speicheroperand 2

* Obwohl Register im Grunde genommen auch ein Teil des Hauptspeichers sind, unterliegen sie hier einer gesonderten Betrachtungsweise. Die allgemeinen Hauptspeicherplätze dienen im Gegensatz zu den Registern zur Speicherung des Programmes (siehe Abb. 75).

spricht man von einem sogenannten SS-Instruktionsformat.
Jetzt liegt natürlich die Vermutung nahe, daß es auch noch andere Instruktionsformate gibt und diese Vermutung wird sich auch sogleich bestätigen:
Denken Sie nur einmal an die Befehle AR, SR, MR oder DR, die reine Registerbezeichnungen als Operanden nach sich ziehen:

Operation Registeroperand 1,Registeroperand 2

Es ergibt sich fast von selbst, daß hier ein RR-Instruktionsformat vorliegt.
Anders verhält es sich bei den Operationen CLI bzw. MVI, in denen neben einem Speicher- auch ein Direktoperand auftritt.

Operation Speicheroperand 1,Direktoperand 2

Nichts ist naheliegender, als in diesem Fall von einem SI-Format zu sprechen.
Außerdem gibt es noch das RX-Format, das unter anderem der LA-Instruktion zugrunde liegt. Vielleicht erinnern Sie sich noch (Kapitel 12), wozu die LA-Instruktion dienen kann: Zum Laden der Adresse eines symbolisch benannten Feldes in ein Register, zum Laden einer Zahl in ein Register oder zur Erhöhung eines Registerinhaltes um einen angegebenen Wert.

```
LA      9,TAB
LA      8,56
LA      8,56(8,0)*
```

In allen diesen Beispielen tritt durchweg als Operand 1 eine Registernummer auf. Etwas schwieriger ist es, beim Operanden 2 eine gewisse Einheitlichkeit festzustellen — aus diesem Grund sei auch die Antwort gleich vorweggenommen: Im Operanden 2 tritt in versteckter, teilweise nicht erkennbarer Form, neben einem Speicheroperanden ein sogenanntes Indexregister auf.

Operation Registeroperand 1, Speicheroperand 2 mit Indexregister.

(Nähere Erläuterung dazu finden Sie am Ende dieses Kapitels. Hier mag zunächst der Hinweis genügen, daß ein solches Indexregister hauptbestimmend für das RX-Format ist).
Vollständigkeitshalber sei auch das RS-Format erwähnt. Diesem RS-Format sind ganz spezielle Instruktionen zugeordnet, deren Behandlung aber über den Rahmen dieses Buches hinausgehen würde. Es läßt sich aber bereits vermuten, daß diese Instruktionen von der Art

Operation Registeroperand 1, Speicheroperand 2

sein müssen. So wird zum Beispiel durch die Instruktion

* In der Instruktion LA 8,56(8,0) fungiert das Register 8 (unterstrichen) als Indexregister.

SLA* 5,OTTO

der Inhalt des Registers 5 um so viele Positionen nach links verschoben, wie die niedrigsten sechs Bits der Adresse des Feldes OTTO angeben. Aber diesen speziellen Hinweis darf man getrost wieder vergessen.
So gehört jeder der weit über hundert bekannten ASSEMBLER-Befehle zu einem der fünf Instruktionsformate (SS, RR, SI, RX, RS).
Ein Blick in ein echtes Programm würde diese Tatsache noch augenscheinlicher machen. Im Kapitel 9 wurde bereits ausführlich darauf hingewiesen, daß jedes symbolische Programm vor seiner eigentlichen Ausführung zunächst einmal umgewandelt werden muß. Denn letzten Endes kann sich der Computer nur nach absoluten Hauptspeicheradressen oder Registernummern orientieren. Wenn man die Register einmal außer acht läßt, sind es also mehr die symbolischen Bezeichnungen, die eine Umwandlung erforderlich machen. In einer echten Insstruktion sind die symbolischen Bezeichnungen dann auch durch die entsprechendenden absoluten Hauptspeicheradressen ersetzt, so wie es in Abb. 109 an einem Beispiel zu erkennen ist.

mnemonischer Befehlsschlüssel	Operand 1	Operand 2
MVC	AUSGAB(6),KUN	

Symbolische ASSEMBLER-Instruktion

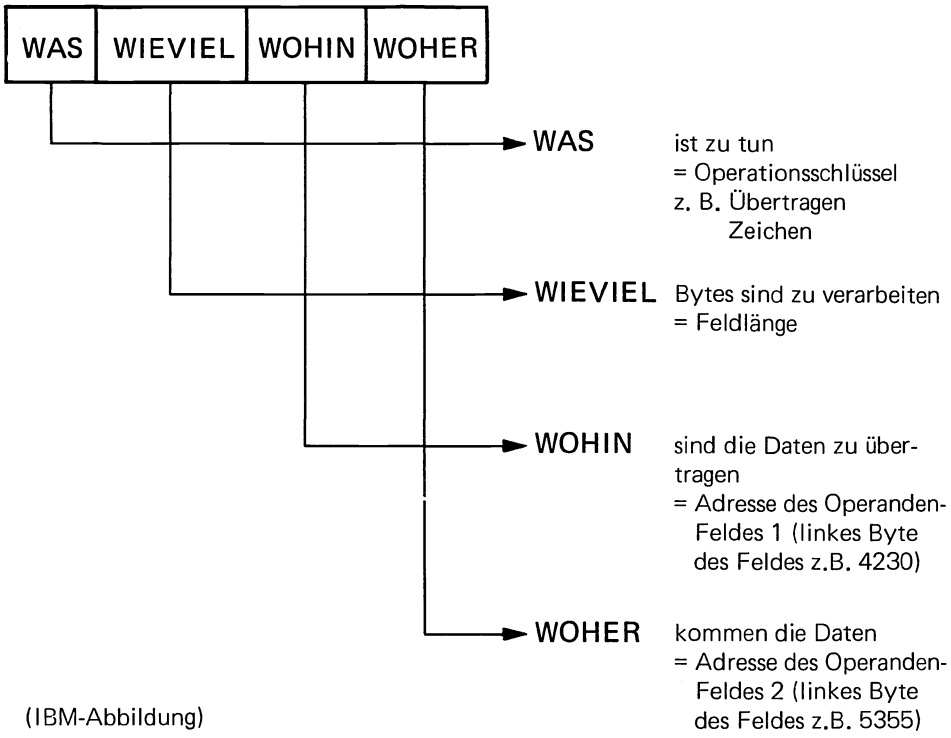
Operationscode	Operand 1	Operand 2
D2	4230	5355

Echte Instruktion (Maschinenbefehl) in vereinfachter Darstellung

Abb. 109

Der Maschinenbefehl zeigt ebenso wie die ASSEMBLER-Instruktion eine Aufgliederung in einen Befehlsschlüssel (Operationscode) und zwei Operanden. Die Angaben sind jetzt allerdings viel nüchterner; sie sind auf die Elektronik des Computers zugeschnitten: Aus dem MVC-Befehlsschlüssel wurde der Operationscode D2, aus den symbolischen Feldnamen bloße Adressen. Unter Berücksichtigung einer Längenangabe ließe sich schließlich der allgemeine Aufbau einer Instruktion wie folgt (Abb. 110) angeben:

* SLA (Shift left single = Links-Shift)



(IBM-Abbildung)

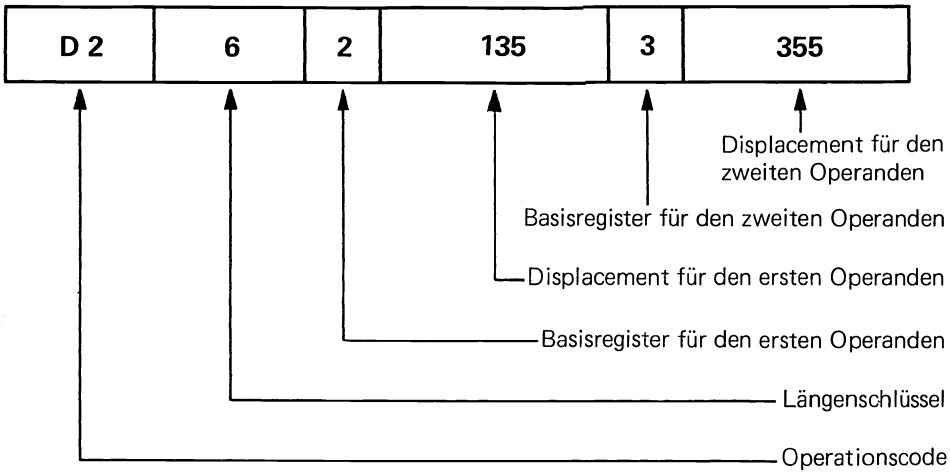
Grundsätzlicher Aufbau einer Instruktion

Abb. 110

Die vereinfachte Darstellung der echten Instruktion in Abb. 109 läßt natürlich noch einige Fragen offen. In Wirklichkeit sind nämlich die absoluten Hauptspeicheradressen gar nicht in dieser expliziten Form vorhanden. Anzutreffen sind für jeden Operanden lediglich die Angaben eines

Displacements (D)
und eines Basisregisters (B)

Als Displacement erscheint immer eine Zahl, als Basisregister stets die Nummer eines Registers. Aus diesen Hinweisen ermittelt sich der Computer von selbst die absolute Kernspeicheradresse, indem er zum Inhalt des Basisregisters das Diplacement addiert.
Während des Umwandlungslaufes werden symbolische Feldnamen, wie AUSGAB und KUN , in entsprechende D- und B-Parameter aufgegliedert.
Angenommen, das echte Programm wäre in den Hauptspeicher geladen worden: Eine echte Instruktion reiht sich dann an die andere. Wie eine solche echte Instruktion aussehen kann, zeigt Abb. 111: Die Instruktion hat eine ganz bestimmte Länge und einen ganz bestimmten Aufbau.



Aufbau einer echten Instruktion mit dem SS-Format

Abb. 111

Bei deren Ausführung stellt sich der Computer wegen des Operationscodes D2 auf Übertragung ein. Die Längenangabe 6 deutet an, wie groß der zu übertragende Bereich ist. Jetzt muß der Computer nur noch die Adresse bestimmen, bei der das empfangene und das abgebende Feld beginnt (Abb. 112). Dabei wird für den ersten Operand zum Inhalt des Basis-Registers 2 (= 4095) das Displacement 135 addiert:

```
  4 095
+   135
= 4 230   Adresse, bei der das empfangende Feld (AUSGAB)
          beginnt
```

Für den zweiten Operanden bezieht sich der Computer auf das (Basis-)Register 3 (mit dem Inhalt von 5 000) und das Displacement 355:

```
  5 000
+   355
= 5 355   Adresse, bei der abgebende Feld (KUN) beginnt
```

Jetzt steht der Ausführung des Übertragungsbefehles nichts mehr im Wege. Es mag allerdings noch unklar geblieben sein, wer wohl den Basisregistern immer den richtigen Inhalt zuordnet: Der Computer tut es von selbst, er ist darin absolut sicher *).

* Vor dem Umwandlungslauf wird der Computer angewiesen, welche Register als Basisregister zu verwenden sind.

Die Stellung der Basisregister und Displacements etc. innerhalb einer echten Instruktion entspricht bestimmten Vorschriften. Für jedes Instruktionsformat gibt es gesonderte Bestimmungen; grundsätzlich haben aber alle Instruktionen gleichen Formats auch den gleichen Aufbau.
Zur detaillierten Darstellung der einzelnen Instruktionsformate verwendet man folgende Abkürzungen:

L	als Längenschlüssel,
B	für ein Basisregister
D	für ein Displacement
X	für ein Indexregister,
I	für einen Direktoperanden,
R	für ein Register

und die Indizes 1 und 2 für den ersten bzw. zweiten Operanden.
Allen Instruktionen des SS-Formates, dazu gehört auch die Übertragungsanweisung in Abb. 111, liegt demnach folgender Aufbau * zugrunde (Abb. 112):

D 2	6	2	135	3	355

ausgewiesener Inhalt des Registers 2 4095
ausgewiesener Inhalt des Registers 3 5000

Beispiel einer echten Übertragungs-Instruktion

Register 2	Register 3
4095	5000

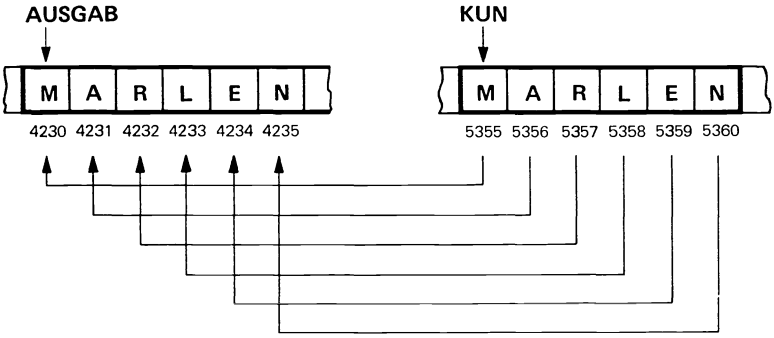


Abb. 112

* In bestimmten Fällen erfolgt noch eine Aufgliederung des Längenschlüssels in L1 = Länge für den ersten Operanden und L2 = Länge für den zweiten Operanden.

In Abb. 113 ist der Aufbau sämtlicher Instruktionsformate erfaßt. Darin treten mitunter R-, X- und I-Parameter als neue Größen auf.
Als R-Parameter erscheint immer eine Registernummer, die auch in der symbolischen Instruktion angegeben wird. Wollte man zum Beispiel den Inhalt des Registers 4 zu dem

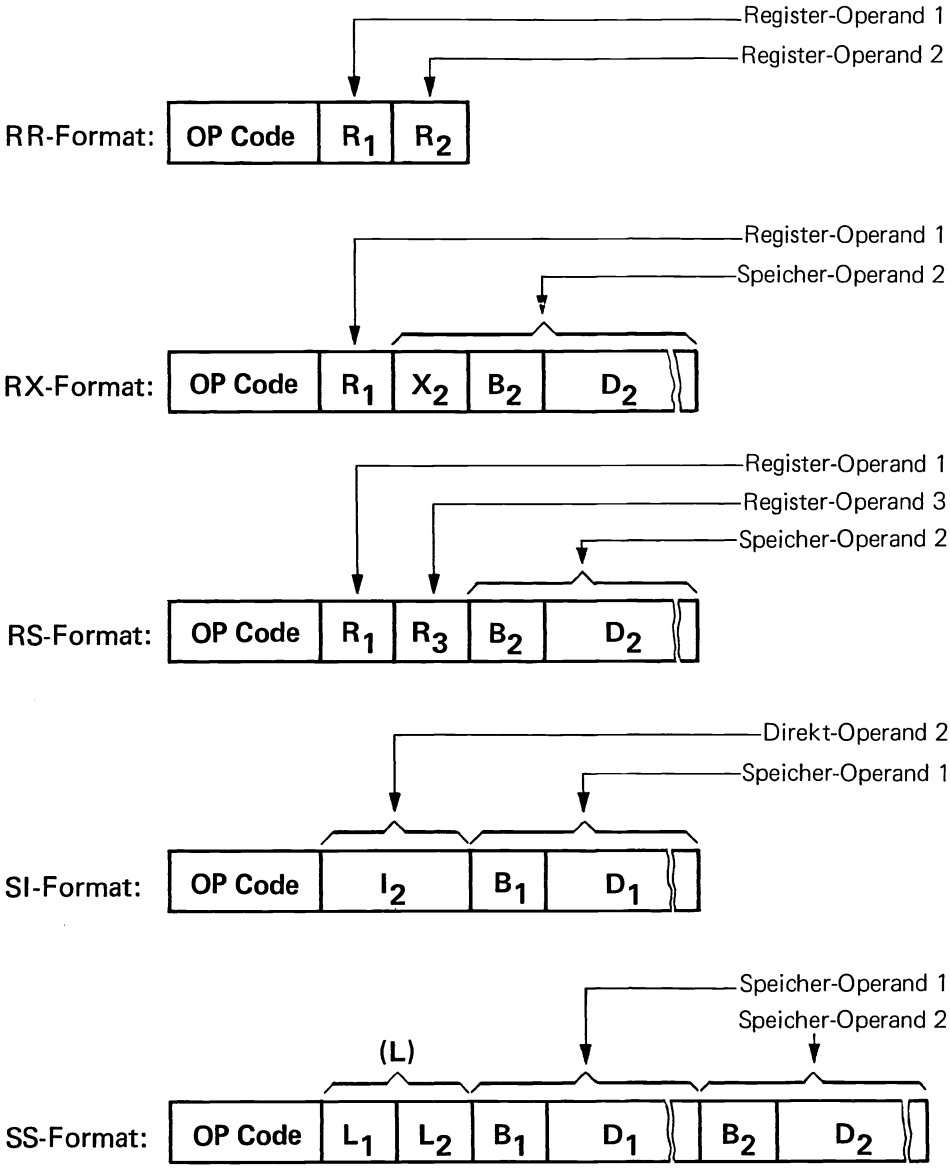


Abb. 113

des Registers 3 addieren, so wäre dafür im symbolischen Programm der RR-Format-Befehl

```
AR        3,4
|         | └── Register-Operand 2
|         └──── Register-Operand 1
└──────────── mnemonsicher Befehlsschlüssel
```

erforderlich.
Nach der Umwandlung sieht die entsprechende echte Instruktion dafür folgendermaßen aus:

```
1A        3 4
|         | └── Register-Operand 2
|         └──── Register-Operand 1
└──────────── Operationscode
```

Daß beim RS-Format plötzlich auch ein Registeroperand 3 auftritt *), soll nicht weiter verwundern. In entsprechenden symbolischen Instruktionen werden manchmal auch wirklich drei Operanden vermerkt. Dieser Umstand ist aber als Ausnahmefall in der Zweiadress-Programmierung zu werten. Anhand eines Beispieles ist schnell geklärt, welche Bedeutung einem solchen dritten Registeroperanden beizumessen ist.:

```
STM       5,7,REGAD
          | | └──── Speicher-Operand 2
          | └────── Register-Operand 3
          └──────── Register-Operand 1
```

Durch den Befehl STM (Store multiple = Speichern mehrfach) wird der Inhalt mehrerer aufeinanderfolgender Register – in diesem Falle der Register 5, 6 und 7 – in dem Kernspeicherfeld REGAD gespeichert.
Welche Bewandtnis hat aber nun das Indexregister X im RX-Format? Es dient dazu, eine Adresse zu indizieren, einen zusätzlichen Einfluß auf die Adressbildung des zweiten Operanden zu nehmen. Dieser Hinweis ist so zu verstehen, daß zum Inhalt des Basisregisters B2 nicht nur das Displacement D2, sondern auch noch der Inhalt des Indexregisters X2 addiert wird. Es bleibt dem Programmierer überlassen, im symbolischen Programm einem der Register die Rolle eines Indexregisters zuzugestehen und diesem einen entsprechenden Inhalt zuzuweisen.
Jetzt ist nur noch der I-Parameter unerwähnt geblieben. An dessen Stelle erscheint immer ein Direkt-Operand. Es handelt sich dabei um den gleichen Direktoperanden, wie er auch vor der Umwandlung in der entsprechenden symbolischen Instruktion (MVI oder CLI) anzugeben ist.
Um die vorstehenden Erläuterungen noch etwas zu untermauern, sind in Abb. 114 für jedes der fünf Instruktionsformate eine symbolische und eine dazugehörige echte Instruktion angegeben. (Die Werte für die Displacements und die Nummern für die Basis-

* Bei manchen Instruktionen des RS-Formates entfällt in der symbolischen Schreibweise die Angabe eines Register-Operanden 3, so auch bei dem SLA-Befehl.

	symbol. Instruktion	echte Instruktion
RR-Format	**AR** *7,8*	**1A** *78*
RX-Format	**LA** *6,10(6,0)*	**41** *660* *10*
RS-Format	**STM** *5,7*,REGAD	**90** *573* 6
ST-Format	**MVI** OTTO,*C'G'*	**92** *C*75 19
SS-Format	**CLC** EMIL*(4)*,FRIDA	**D5** *4*71 88 20

Abb. 114

und Indexregister wurden willkürlich gewählt).
Die echten Instruktionen beanspruchen im Hauptspeicher je nach Format mehr oder weniger Platz; die des RR-Formats benötigen jeweils zwei Bytes (= vier Halbbytes); für das RX-, RS- und SI-Format sind jeweils vier Bytes (= acht Halbbytes) und für das SS-Format sechs Bytes (= 12 Halbbytes) erforderlich. Ließe man sich echte Instruktionen halbbyteweise (also hexadezimal) ausdrucken, so ergäbe sich beispielsweise eine Darstellung wie in der rechten Spalte von Abb. 114. (Die fettgedruckten Angaben beziehen sich auf den Operationscode, die in Kursiv gedruckten Angaben der echten und symbolischen Instruktion entsprechen einander, während die den symbolischen Feldnamen zugeordneten Basisregister und Displacements nur in halbfett erscheinen. Das Indexregister in der LA-Instruktion des RX-Formats ist in fetter Kursivschrift hervorgehoben).
Die vorstehenden Bemerkungen dieses Kapitels beziehen sich in erster Linie auf echte Maschinenbefehle. Daß dabei zwangsweise auch die einzelnen Parameter (R, B, D, X, L, I) der verschiedenen Instruktionsformate erwähnt werden, ist nicht nur für das Lesen eines echten Programmes, sondern in hohem Maße auch für die symbolische Programmierung wichtig. Denn auch in einer symbolischen Instruktion dürfen echte Parameter angegeben werden, selbstverständlich unter Beachtung bestimmter Vorschriften (Abb. 115).
So muß zum Beispiel die MVC-Instruktion

```
MVC     SUSU+2(3),MARI
```

dem formellen Aufbau

```
MVC     D1(L1,B1),D2(B2)
```

D1 = Displacement für Operand 1
L1 = Längenschlüssel für Operand 1
B1 = Basisregister für Operand 1
D2 = Displacement für Operand 2
B2 = Basisregister für Operand 2

Instruktions-format	symbol. ASSEMBLER-Instruktion in Parameter-Schreibweise		Bedeutung	Operations-Code* der echten Instruktion
RR	AR	R1,R2	Addiere	1A
SS	CLC	D1(L,B1),D2(B2)	Vergleiche Zeichen	D5
SI	CLI	D1(B1),I2	Vergl. Dir.-Zeichen	95
RX	CVB	R1,D2(X2,B2)	Umwandeln in binär	4F
RX	CVD	R1,D2(X2,B2)	Umwandeln in dezimal	4E
RR	DR	R1,R2	Dividiere	1D
RX	LA	R1,D2(X2,B2)	Lade Adresse	41
SI	MVI	D1(B1),I2	Übertrg. Dir.-Zeichen	92
SS	MVC	D1(L,B1),D2(B2)	Übertrage Zeichen	D2
RR	MR	R1,R2	Multipliziere	1C
SS	PACK	D1(L1,B1),D2(L2,B2)	Packe	F2
RS	SLA	R1,D2(B2)	Links Shift	8B
RS	STM	R1,R3,D2(B2)	Speichern mehrfach	90
RR	SR	R1,R2	Subtrahiere	1B
SS	UNPK	D1(L1,B1),D2(L2,B2)	Entpacke	F3
SS	AP	D1(L1,B1),D2(L2,B2)	Addiere dezimal	FA
SS	DP	D1(L1,B1),D2(L2,B2)	Dividiere dezimal	FD
SS	MP	D1(L1,B1),D2(L2,B2)	Multipliziere dezimal	FC
SS	SP	D1(L1,B1),D2(L2,B2)	Subtrahiere dezimal	FB

* halbbyteweise Darstellung

Abb. 115

genügen. Wollte man unter Beachtung dieser Vorschrift anstelle der Instruktion

```
MVC     SUSU+2(3),MARI
```

lieber eine Schreibweise mit echten Parametern bevorzugen, so wäre folgender Ausdruck denkbar

```
MVC     2(3,6),0(8)
```

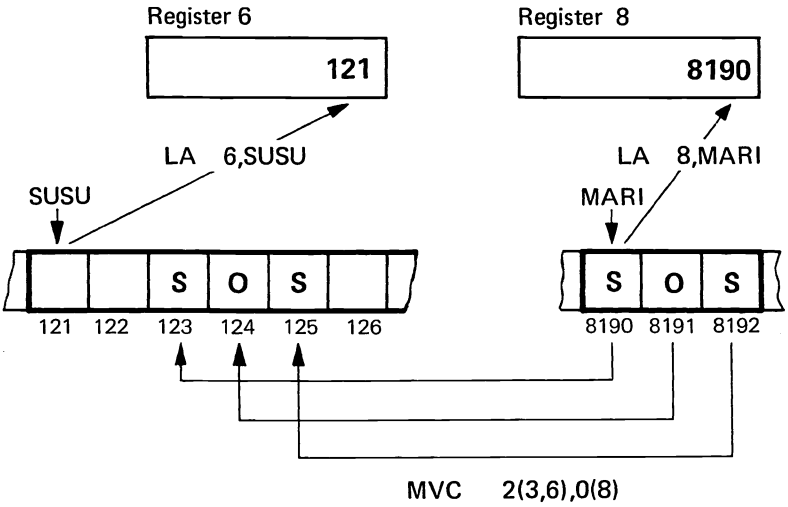
Vor Ausführung dieses Befehles müßten allerdings die als Basisregister verwendeten Register 6 und 8 mit den absoluten Adressen der Felder SUSU und MARI geladen werden (Abb. 116):

```
LA      6,SUSU
LA      8,MARI
```

Auf diese Registerstände (das Register 6 enthält die absolute Hauptspeicheradresse 121 des Feldes SUSU, das Register 8 die absolute Hauptspeicheradresse 8190 des Feldes MARI) bezieht sich dann die nachfolgende Instruktion

```
MVC     2(3,6),0(8)
```

Erläuterung: In der Länge von L1 = 3 Bytes wird übertragen (siehe Abb. 116):



entweder	oder
MVC SUSU+2(3),MARI	LA 6,SUSU LA 8,MARI MVC 2(3,6),0(8)

Abb. 116

Von der Adresse	0	(Displacement D2)
	+ 8190	(Inhalt des Basisregisters 8)
	= 8190	
zur Adresse	2	(Displacement D1)
	+ 121	(Inhalt des Basisregisters 6)
	= 123	

Dem Programmierer sind die absoluten Adressen der Felder SUSU und MARI unbekannt. Der Computer hingegen weiß vom Umwandlungslauf her, welchen Platz diese Felder im Hauptspeicher einnehmen. Demnach soll das Feld SUSU bei der Adresse 121, das Feld MARI bei der Adresse 8190 beginnen.
Aus diesem Grunde ist es möglich, den Computer durch eine LA-Instruktion anzuweisen, diese Adressen in Registern zur Verfügung zu stellen. Nach Ablauf der Instruktionen

```
LA    6,SUSU
LA    8,MARI
```

steht also im Register 6 die Adresse 121, im Register 8 die Adresse 8190. Da sich die

Übertragung auf eine Länge von drei Bytes beziehen soll, werden die Adressen des abgebenden und empfangenden Feldes selbsttätig immer wieder um 1 erhöht, bis der gesamte Übertragungszyklus abgeschlossen ist. Die Übertragung vollzieht sich also in drei Phasen:

1. Übertragung des Inhaltes der Speicherstelle 8190 zur Speicherstelle 123
2. Übertragung des Inhaltes der Speicherstelle 8191 zur Speicherstelle 124
3. Übertragung des Inhaltes der Speicherstelle 8192 zur Speicherstelle 125.

Alle diese Vorgänge sind schematisch in Abb. 116 dargestellt.
Zunächst mag die Verwendung von Displacement und Basisregistern in symbolischen Instruktionen noch sehr umständlich erscheinen. Doch bereits im nächsten Kapitel wird gezeigt, welche Vorteile damit verbunden sein können.
In der tabellarischen Aufstellung nach Abb. 115 sind die allgemeinen Vorschriften für den Aufbau symbolischer Instruktionen mit echten Paramtern wiedergegeben.
Die Parameter D und B bezeichnen grundsätzlich immer einen Speicheroperanden, für den der Programmierer ebensogut auch einen symbolischen Namen vergeben kann. Für MVC-Befehle sind also Schreibweisen wie

```
MVC     AUSGAB,KARBER
MVC     OTTO(3),IDA
MVC     EMIL+5(2),FRIDA+9
MVC     EMIL+5,LOLA
MVC     9(7,3),EDDA
MVC     10(7,3),9(8)
MVC     BALIM(3),6(5) usw.
```

möglich. Unterbleibt eine explizite Längenangabe im ersten Operanden, so bezieht sich die Übertragung immer auf die (definierte) Länge des ersten Operandenfeldes. Doch solche detaillierten Hinweise sollen der Spezialliteratur vorbehalten bleiben.
Hier genügt eine allgemeine Kenntnis der einzelnen Instruktionsformate.
Zur Wiederholung sei nochmals das Wichtigste zusammengefaßt: Die einzelnen Instruktionen unterliegen einem ganz bestimmten Aufbau, der durch die Instruktionsformate festgelegt ist. In den Operanden echter Instruktionen treten sogenannte echte Parameter (Basisregister, Displacement, Indexregister, Längenschlüssel, Direktoperanden oder bloße Registernummern) in einer typischen Reihenfolge auf (Abb. 113). Diese Parameter dürfen aber auch neben den symbolischen Feldnamen in symbolischen ASSEMBLER-Instruktionen angegeben werden; allerdings nur unter Berücksichtigung einer entsprechenden symbolischen Parameterschreibweise (Abb. 115). Die Vorteile einer solchen Möglichkeit werden im Kapitel 16 offenbar.

16. Tabellen-Lesen

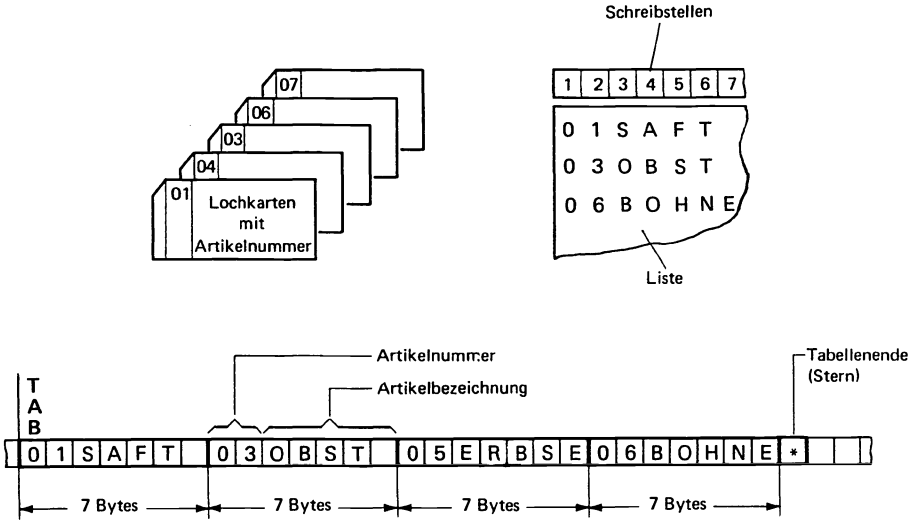
Das Tabellen-Lesen gehört ebenso zu den Grundlagen des Programmierens wie der Gruppenwechsel und die Verwendung eines programmierten Schalters. Der Begriff des Tabellen-Lesens läßt sich wiederum am besten anhand eines Beispieles erklären:
Angenommen, es lägen Auftrags-Lochkarten zur Verarbeitung vor, in denen in den Spalten 3 bis 4 eine Artikelnummer abgelocht ist. In allen übrigen Spalten 1 bis 2 und 5 bis 80 sind andere wichtige Angaben enthalten, jedoch keine Artikelbezeichnung, wie Saft oder dergleichen.
Und auf diese Artikelbezeichnung, für die in der Auftrags-Lochkarte kein Platz ist, kommt es gerade an. Denn die Aufgabe lautet, auf einer Liste neben der Artikelnummer auch die zugehörige Artikelbezeichnung auszudrucken.
Wie kann aber nun im Programm einer Artikelnummer die zugehörige Artikelbezeichnung zugeordnet werden? Gäbe ein Fachmann die Antwort, so fielen wahrscheinlich nur die beiden Worte: „per Tabelle". Doch was verbirgt sich dahinter?
Eine Tabelle ist ja ohnehin nichts weiter als eine fortlaufende Aufstellung gleichartiger Angaben. In unserem Falle wäre es also angebracht, diese Aufstellung auf die Artikelnummern und die zugehörigen -bezeichnungen abzustellen, etwa in der Art

```
01SAFT
03OBST
05ERBSE
06BOHNE
```

Liegt beispielsweise die Artikelnummer 05 vor, so ließe sich ohne Schwierigkeit aus der Tabelle die gewünschte Bezeichnung ERBSE heraussuchen, ein Vorgang, den man einfach als Tabellen-Lesen bezeichnet.
Um den Computer mit dem Lesen einer Tabelle betrauen zu können, muß diese zunächst einmal in den Kernspeicher „gelangen" (Abb. 117). Zu diesem Zweck macht man unter der Rubrik Definitonen des Programmes folgende Angaben:

```
TAB     DC  CL7'01SAFT '
        DC  CL7'03OBST '
        DC  CL7'05ERBSE'
        DC  CL7'06BOHNE'
        DC  CL1'*'
```

Nach dem Programmladen erscheint die Tabelle TAB in der in Abb. 117 angegebenen Form im Hauptspeicher. Die Programmierung des Tabellen-Lesens läßt sich besonders einfach durchführen, wenn die einzelnen Tabellenabschnitte die gleiche Länge (in diesem Falle 7 Bytes) aufweisen. In den ersten beiden Stellen jedes Tabellenabschnittes steht die Artikelnummer, in den nachfolgenden fünf Stellen die entsprechende Artikelbezeichnung.
Nach dem Einlesen der ersten Auftrags-Lochkarte steht deren Artikelnummer in der 3. und 4. Stelle des Karteneinlesebereiches KARBER. Es ist nun Aufgabe des Programmierers, entsprechende Befehle vorzusehen, mit denen die gewünschte Artikelbezeichnung aus der Tabelle herausgesucht und in den Druckausgabebereich AUSGAB übertragen werden kann.
Die Lösung dieser Aufgabe ist recht simpel: Man durchforstet die Tabelle abschnittweise



Artikel-Tabelle im Hauptspeicher

Abb. 117

nach der vorliegenden Artikelnummer. Sollte einmal der Fall auftreten, daß ein gesuchter Artikel in der Tabelle nicht enthalten ist, so muß die Suche spätestens am Tabellenende (beim Stern-Zeichen) abgebrochen werden.
Deshalb sollte auch vor jedem neuen Tabellensuchschritt zunächst danach gefragt werden, ob etwa das Tabellenende (der Stern) bereits erreicht wurde. Bei der Programmierung des Tabellen-Lesens bezieht man sich zweckmäßigerweise zunächst auf die Adresse des Tabellenanfangs und erhöht diesen Wert mit jedem Suchschritt um die Länge eines Tabellenabschnittes, also um 7 Bytes.
Im Blockdiagramm nach Abb. 118 sind die einzelnen Instruktionen für eine solche Lösung aufgeführt:
Bei jedem Tabellensuchschritt wird die Artikelnummer der gerade eingelesenen Auftrags-Lochkarte mit der Artikelnummer eines Tabellenabschnittes verglichen. In der Vergleichsinstruktion treten als Operanden nicht etwa symbolische Feldnamen, sondern Registerbezeichnungen auf. In diesen Registern sind die Adressen der zu vergleichenden Kernspeicherplätze gespeichert.
Für die Adresse der dritten Stelle des Karteneinlesebereiches (KARBER+2) – dort beginnt die Artikelnummer der Auftrags-Lochkarte – ist das Register 5 vorgesehen. Das Register 6 soll zunächst die Adresse aufnehmen, bei der die Tabelle TAB mit der ersten Artikelnummer ihren Anfang nimmt.
Jetzt kann bereits durch Verweis auf die Adressen in den Registern 5 und 6 (bei einem Längenschlüssel von 2 Bytes) der erste Tabellensuchschritt eingeleitet werden.
Stellt sich heraus, daß die verglichenen Artikelnummern nicht identisch sind, so ist die Tabelle weiter zu durchsuchen. Die Adresse im Register 6 wird um den Wert 7 (also um die Länge eines Tabellenabschnittes) erhöht. Und über die Instruktion STERN kehrt man wieder zu dem Befehl

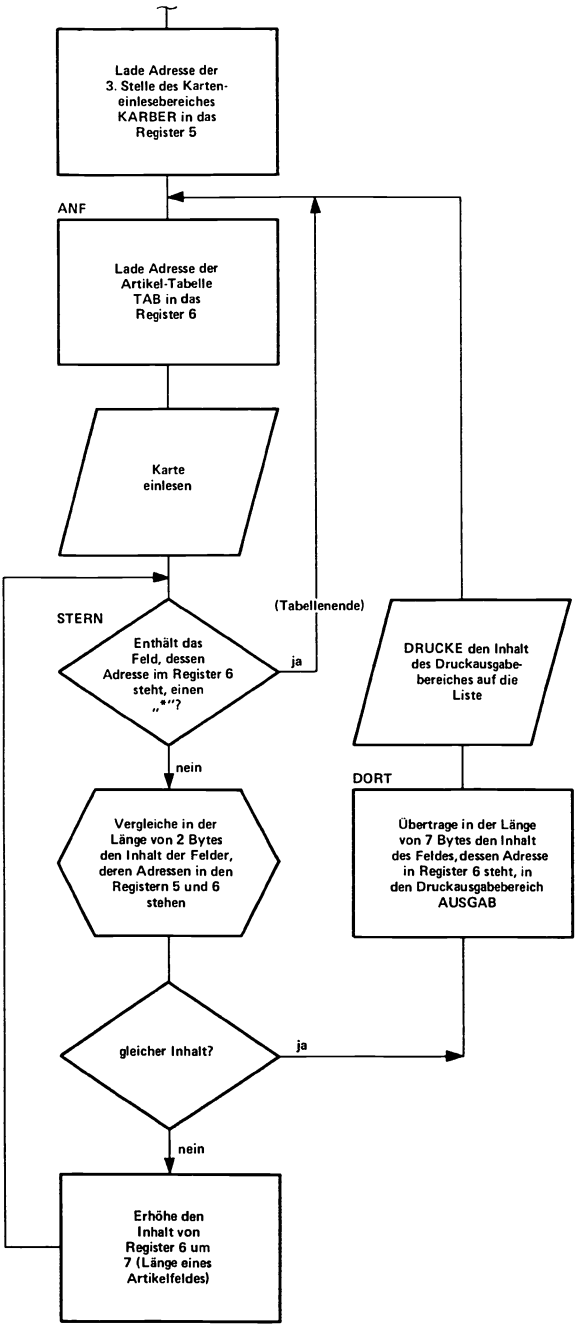

Lade Adresse der 3. Stelle des Karten-einlesebereiches KARBER in das Register 5
ANF
Lade Adresse der Artikel-Tabelle TAB in das Register 6
Karte einlesen
STERN
Enthält das Feld, dessen Adresse im Register 6 steht, einen "*"?
ja
(Tabellenende)
nein
Vergleiche in der Länge von 2 Bytes den Inhalt der Felder, deren Adressen in den Registern 5 und 6 stehen
gleicher Inhalt?
ja
nein
Erhöhe den Inhalt von Register 6 um 7 (Länge eines Artikelfeldes)
DRUCKE den Inhalt des Druckausgabe-bereiches auf die Liste
DORT
Übertrage in der Länge von 7 Bytes den Inhalt des Feldes, dessen Adresse in Register 6 steht, in den Druckausgabebereich AUSGAB

Abb. 118

Vergleiche in der Länge von 2 Bytes den Inhalt der Felder, deren Adressen in den Registern 5 und 6 stehen

zurück.

Auf diese Art und Weise wird jetzt die Artikelnummer der eingelesenen Lochkarte mit der des nächsten Tabellenabschnittes verglichen. Sollte sich dabei eine Identität ergeben, so brauchten nur noch Artikelnummer und Artikelbezeichnung dieses Tabellenabschnittes (dessen Adresse im Register 6 steht) in einer Länge von 7 Bytes in den Druckausgabebereich AUSGAB übertragen zu werden.

Das Durchsuchen der Tabelle kann aber auch erfolglos sein. Dieser Fall wird immer dann auftreten, wenn in der Auftrags-Lochkarte fälschlicherweise eine Artikelnummer abgelocht ist, die in der Tabelle gar nicht auftritt. Zunächst würde im Programm ein Tabellenabschnitt nach dem anderen unter ständiger Erhöhung der Adresse im Register 6 nach dieser Artikelnummer durchsucht werden. Um zu verhindern, daß sich diese Suche über das Tabellenende hinaus erstreckt, bedient man sich der Instruktion

STERN Enthält das Feld, dessen Adresse im Register 6 steht, einen „*"?

(Zur Kennzeichnung des Tabellenendes wurde nach dem allerletzten Tabellenabschnitt ein Sternzeichen definiert).

Durch diese Frage STERN wird – nach Erhöhung des Inhaltes des Registers 6 – entschieden, ob das Tabellenende bereits erreicht ist oder nicht. Im ersten Fall ist die gesuchte Artikelnummer in der Tabelle nicht enthalten; eine neue Lochkarte kann eingelesen werden.

Bevor jedoch der Suchzyklus für die Artikelnummer einer neuen Auftrags-Lochkarte beginnen kann, ist wieder die Anfangsadresse der Tabelle TAB in das Register 6 zu laden.

Bei der Codierung im Zeilenschema (Abb. 119) werden erstmals ASSEMBLER-Instruktionen in Parameterschreibweise benutzt:

```
CLI     D1(B1),I2
CLC     D1(L,B1),D2(B2)
LA      R1,D2(X2,B2)
MVC     D1(L,B1),D2(B2)
```

Es ist statthaft (Kapitel 15), einen Operanden symbolisch, den anderen durch D-, B-, L- bzw. X-Parameter zu benennen, wie es zum Beispiel in der Instruktion

```
MVC     AUSGAB(7),0(6)
```

geschieht. Im vorausgehenden Kapitel wurde bereits erwähnt, daß die Adreß-Bildung für einen Operanden durch Addition des Displacements D zum Inhalt des Basisregisters B und, falls vorhanden, zum Inhalt des Indexregisters X erfolgt.

Für die Lochkarte ist im Zeilen-Schema der Dateiname KART, für die auf dem Drucker auszugebende Liste der Dateiname DRUCK vorgesehen; der Karteneinlesebereich trägt den Namen KARBER, der Druckausgabebereich den Namen AUSGAB. Und nun verfolgend Sie bitte in Abb. 119, welch unerwartet geringen Aufwand ein programmiertes Tabellenlesen erfordert: Wenige Instruktionen vollziehen einen Suchvorgang, dessen nähere Erläuterungen hier mehrere Seiten füllt.

UNIVAC

ASSEMBLER

Programm TABELLEN-LESEN Programmierer _____ Datum _____ Blatt 01

Name (1–8)	Operation (10–15)	Operand → Bemerkungen (16–71)	Programm / Folge (73–80)
			01
			02
			03
	LA	5,KARBER+2	04
ANF	LA	6,TAB	05
	GET	KART	06
STERN	CLI	0(6),C'*'	07
	BE	ANF	08
	CLC	0(2,5),0(6)	09
	BE	DORT	10
	LA	6,7(6,0)	11
	B	STERN	12
DORT	MVC	AUSGAB(7),0(6)	13
	PUT	DRUCK	14
	B	ANF	15
			16
			17
			18
			19
			20
			21
			22
			23
			24

UC 122 C

Abb. 119/120

17. Sortieren und Mischen

Ein altes Sprichwort besagt:

„Ordnung ist das halbe Leben"

und das gilt beileibe nicht nur für Pedanten. Allerorten lassen sich genügend Beispiele finden, die einem gewissen Ordnungsprinzip huldigen:

Die alphabetische Reihenfolge der Namen im Telefonbuch,
die aufsteigenden Hausnummern in Straßenzügen,
die Einordnung von Bankkunden nach ihrer Kontonummer,
die Festlegung der Autokennzeichen nach bekannten Gesichtspunkten.

Überall tritt als Ordnungsprinzip entweder eine alphabetische, numerische oder alphabetisch-numerische (alpha-numerische) Reihenfolge auf. Sie allein ermöglicht eine zügige, systematische Orientierung; was gäbe es bspws. in Notfällen für ein heilloses Durcheinander, wenn man die Rufnummer seines Arztes vergessen hätte, die jeglicher alphabetischen Ordnung zum Hohn irgendwo im Telefonbuch stünde.

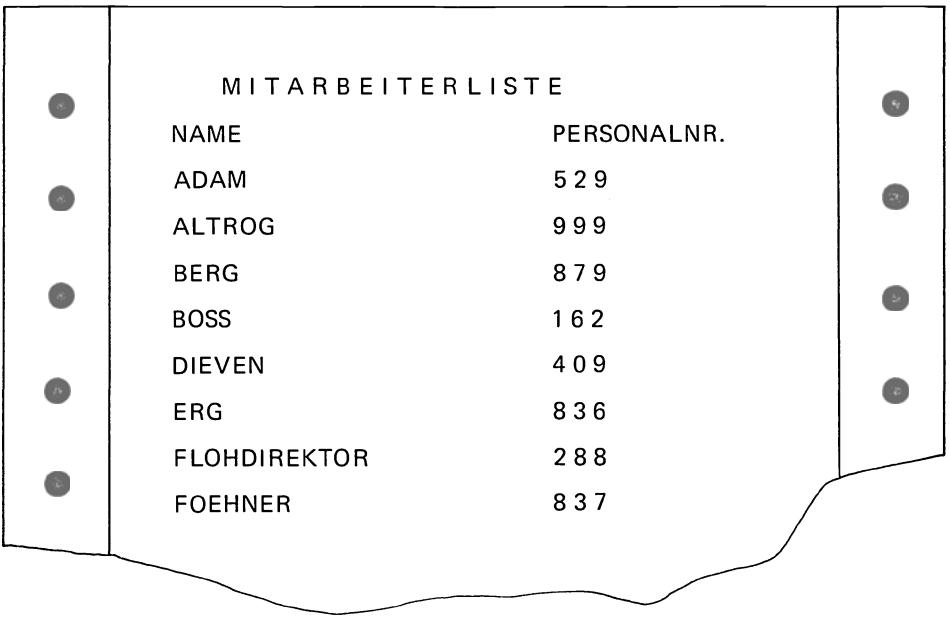
Aus Gründen der Übersichtlichkeit und im Interesse einer wirkungsvollen Organisation finden diese Ordnungsprinzipien auch in der elektronischen Datenverarbeitung Berücksichtigung. Dies wird besonders auf Listen augenscheinlich, die während verschiedener Programmläufe über den Drucker ausgegeben werden:

So erfolgt die Aufstellung der Mitarbeiter eines Betriebes entweder nach ihrer aufsteigenden Personalnummer (Abb. 121) oder in alphabetischer Reihenfolge (Abb. 122).

MITARBEITERLISTE

PERSONALNR.	NAME
162	BOSS
288	FLOHDIREKTOR
291	ZIEHTEN
343	OLGEN
409	DIEVEN
529	ADAM
699	MANN
715	OSTOER
836	ERG

Abb. 121



MITARBEITERLISTE

NAME	PERSONALNR.
ADAM	529
ALTROG	999
BERG	879
BOSS	162
DIEVEN	409
ERG	836
FLOHDIREKTOR	288
FOEHNER	837

Abb. 122

Doch der Vorteil einer solchen Übersichtlichkeit muß erkauft werden: Die Lochkarten, deren Daten während eines Programmlaufes auf einer Liste erscheinen sollen, müssen in einer zuvor entsprechend sortierten Reihenfolge dem Karten-Leser zugeführt werden. Gibt es, Lochkarten der in Abb. 123 gezeigten Feldeinteilung wie in Abb. 121 angegeben, zu listen, so ist es unbedingt erforderlich, sie zunächst nach ihrem Ordnungsbegriff (der numerischen Personalnummer in Spalte 2 bis 4) in die gewünschte

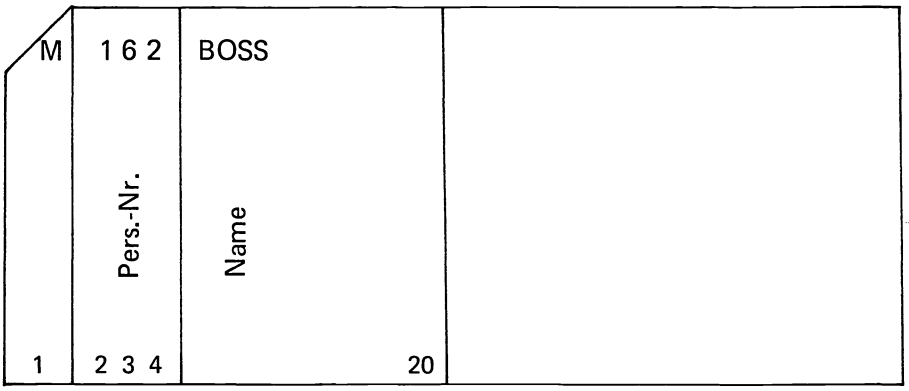


Abb. 123

Reihenfolge zu bringen. Nun wäre es zweifelsohne ein recht mühsames Unterfangen, diese Arbeit „von Hand" vornehmen zu wollen; für Tausende solcher Lochkarten würde dies eine tage – wenn nicht gar wochenlange Arbeit bedeuten.
Einen Ausweg aus diesem Dilemma bot früher die Sortiermaschine, mit deren Hilfe sich ungeordnete Lochkarten in eine gewünschte numerische oder auch alphabetische Reihenfolge bringen ließen: ein übersichtliches aber sehr zeitaufwendiges Unterfangen. Heute dagegen werden Daten fast ausnahmslos durch Sortierprogramme (siehe Kapitel 23) geordnet, die sich auf zunächst ungeordnete Daten in Magnetband- oder Magnetplatten-Einheiten (siehe Kapitel 20 und 21) anwenden lassen.
Der Begriff des Mischens wird zunächst Ihre Gedanken an ein Skatspiel oder an einen Cocktail wachrufen, weniger jedoch einen Zusammenhang mit der Computertechnik vermuten lassen. Und gerade hier erlangt der Mischvorgang eine Bedeutung, die ihn zu einem der Grundpfeiler elektronischer Datenverarbeitung werden läßt.
Während das Mischen eines Skatspieles noch darauf abgestellt ist, die Spielkarten in eine möglichst ungeordnete Reihenfolge zu bringen, sollen in der Datenverarbeitung durch das Mischen Lochkarten nach bestimmten Gesichtspunkten geordnet werden.
Für diesen Zweck stand früher ein sogenannter Karten-Mischer (Abb. 127) zur Verfügung, dessen Arbeitsweise an nachstehenden Beispiel erläutert werden soll:
Ein kleinerer Betrieb hat für seine Rechnungsschreibung eine Kunden-Kartei und eine Bestell-Kartei (Feldeinteilung der Lochkarten nach Abb. 125) angelegt (Abb. 124).
Sämtliche Kunden sind mit ihrer Anschrift und Kundennummer in der Kunden-Kartei erfaßt. Sobald eine Bestellung einläuft, wird unter Angabe der Kundennummer für jeden Artikel eine Bestellochkarte erstellt.
Die Lochkarte innerhalb jedes Kartei-Kastens müssen nach aufsteigender Kundennummer sortiert sein, eine Forderung, die sich mühelos mit einer Sortiermaschine erfüllen läßt.
Sobald der Termin für eine Rechnungsschreibung (per Programm) heranrückt – in manchen Betrieben geschieht dies täglich, in anderen wöchentlich oder monatlich einmal – müssen beide Karteien (die Kundenadreß- und die Bestellochkarten) in einer ganz bestimmten Art und Weise vereinigt (gemischt) werden: Hinter jeder Kundenadreßlochkarte (punktiert) sollen die zugehörigen Bestellochkarten (mit der gleichen Kundennummer wie in der „führenden" Kundenadreßlochkarte) zu liegen kommen (Abb. 127). Auszusortieren sind dabei lediglich jene Adreßlochkarten, deren Kunden keine Bestellung aufgegeben haben.

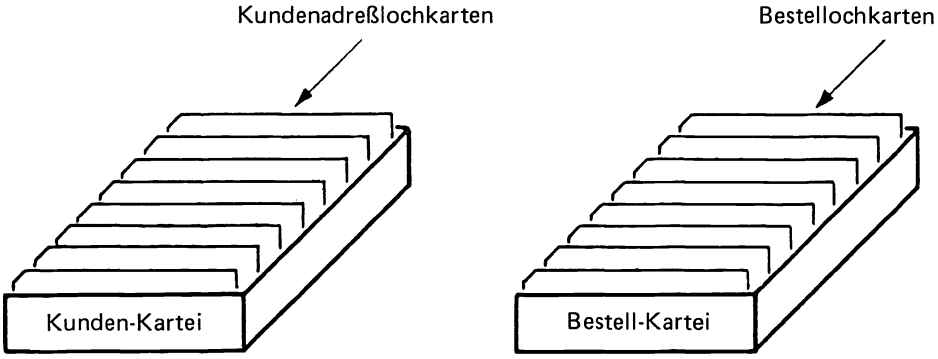
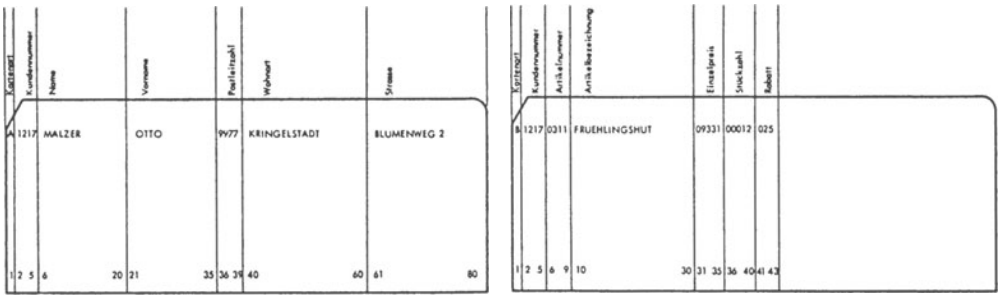


Abb. 124



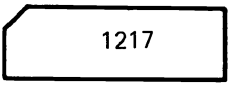
Feldeinteilung

der Kundenadreßlochkarte

der Bestellochkarte

Abb. 125

Kundenadreßlochkarte



Bestellochkarte

Vereinfachte Darstellung der Lochkarten für den Mischvorgang unter Angabe der Kundennummer

Abb. 126

Zur Vereinfachung der Darstellung seien die Lochkarten nur durch Angabe ihrer Kundennummer gekennzeichnet (Abb. 126).
Die maschinelle Vereinigung beider Karteien wird von einem Karten-Mischer selbstätig durchgeführt. Dazu sind die Kundenadreßlochkarten in sein Erstkartenmagazin, die Bestellochkarten in sein Zweitkartenmagazin einzulegen. Nach dem Einstellen der gewünschten Betriebsart

Mischen mit Aussondern unpaariger Erstkarten

kann endlich der START-Knopf gedrückt werden: Die gemischten Lochkarten werden im Fach 2 abgelegt, die auszusondernden (unpaarigen Erstkarten) Adreßkarten (ohne zugehörige Bestellkarten) gelangen in das Fach 1 (Abb. 127).
Das gemischte Kartenpaket (Fach 2) kann anschließend als Eingabedatei für den Programmlauf „Rechnungsschreibung" verwendet werden. Auf eine Adreßlochkarte folgen immer die zugehörigen Bestellochkarten, so daß das Rechnungsformular in der Reihenfolge der eingegebenen Lochkarten — erst die Kundenanschrift dann die Artikelaufstellungen — gedruckt werden kann.
Es sei bemerkt, daß dieses Beispiel nur dazu dient, den Mischvorgang verständlich zu machen. Denn heute werden Karten-Mischer so gut wie nicht mehr verwendet. Ihre Funktion übernehmen ungleich schnellere Mischprogramme, die wie auch Sortierprogramme, auf Band- und Plattendaten angewendet werden.

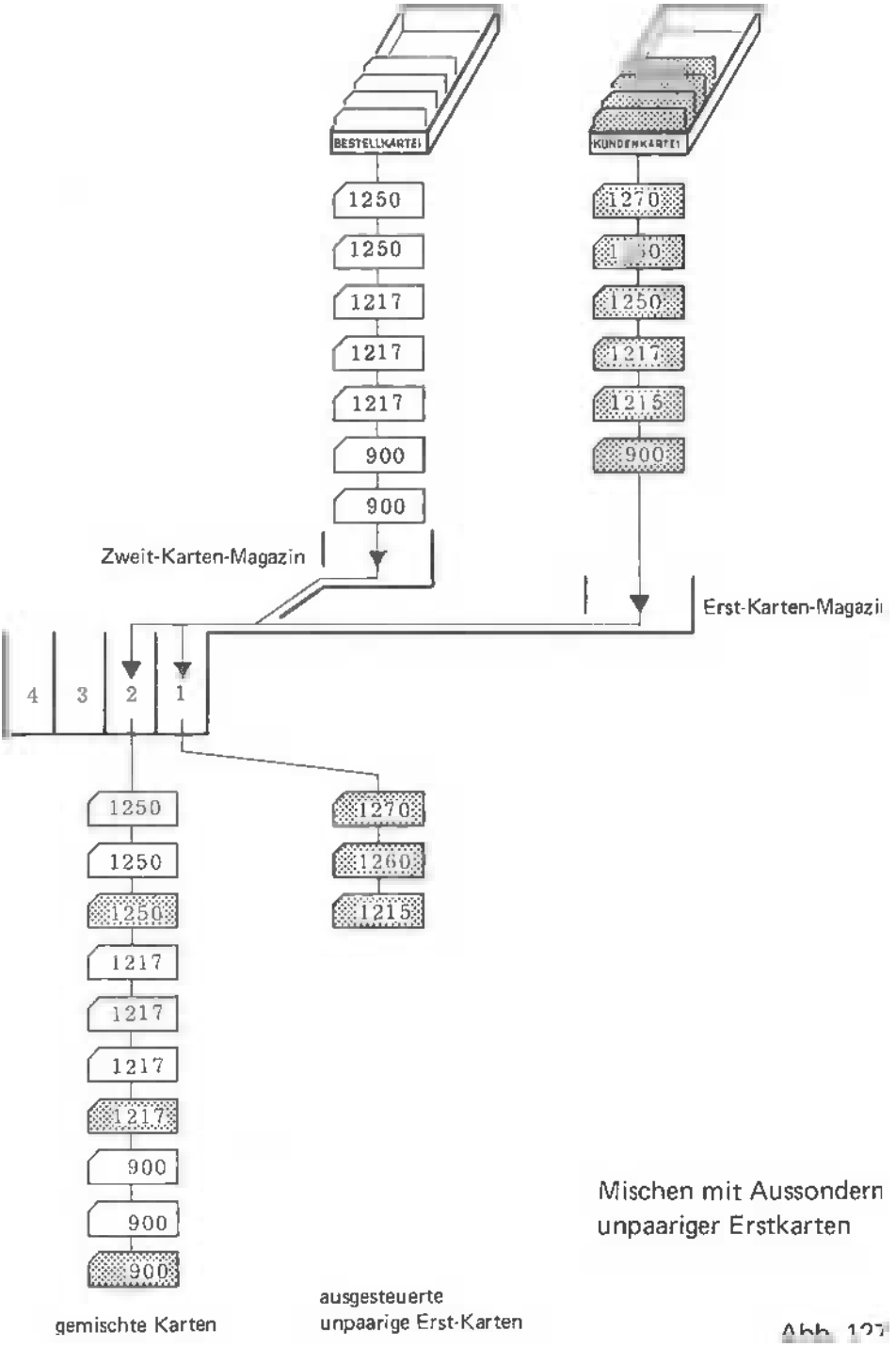

BESTELLKARTEI
KUNDENKARTEI
1250
1250
1217
1217
1217
900
900
1270
1250
1217
1215
900
Zweit-Karten-Magazin
Erst-Karten-Magazi
4
3
2
1
1250
1250
1250
1217
1217
1217
1217
900
900
900
1270
1260
1215
Mischen mit Aussondern
unpaariger Erstkarten
gemischte Karten
ausgesteuerte
unpaarige Erst-Karten

Abb. 127

18. Verschiedene Ein/Ausgabe-Einheiten*)

Von dem „geheimnisvollen Schleier", der anfangs noch die elektronische Datenverarbeitung umgab, ist inzwischen nicht mehr viel übrig geblieben.
Die Daten, um deren Verarbeitung es letzten Endes geht repräsentierten sich zum Beispiel als Preise, Bestellmengen, Rabattsätze und Artikelbezeichnungen. Ihre Verarbeitung erforderte die Durchführung vieler Rechenoperationen (wie Preis x Bestellmenge) und anderer Anweisungen. Zur Lösung einer bestimmten Aufgabe werden die entsprechenden Befehle zu einem Programm zusammengestellt. Der Computer nimmt seine Anweisungen jedoch nur in einer ganz bestimmten Art entgegen, in der Formulierung einer Programmiersprache. Je nach dem vorliegenden Problem, sei es nun kommerziellen, technischen oder rein mathematischen Charakters, kann man sich u. a. entweder für COBOL, FORTRAN, ALGOL, PL/1 oder für den ASSEMBLER entscheiden.
Um den Computer zu bewegen, ein Programm in sein „Gedächtnis" aufzunehmen, wurde dieses mittels Lochkarten in den Hauptspeicher der Zentral-Einheit geladen. Erst dann war der Computer bereit, alle gespeicherten Instruktionen nacheinander auszuführen. Dabei spielten spezielle, elektronische Schaltkreise (Kapitel 13) für Übertragungen, Additionen usw. eine große Rolle.
So erfolgte in der Zentral-Einheit die eigentliche Verarbeitung der Daten. Dabei wurde der Inhalt von Hauptspeicherfeldern entsprechend verknüpft. Die Art der Verknüpfung (ein Vergleich, eine Subtraktion oder eine Addition) war durch den Operationscode der Instruktion bestimmt, die Lage der Kernspeicherfelder durch deren Adressen.
Was für den Computer die Zentral-Einheit darstellt, ist für den Menschen das Gehirn. Dort werden zwar nicht Daten im üblichen Sinne, dafür aber andere Informationen verarbeitet; manchmal zu guten, manchmal allerdings auch zu schlechten Vorsätzen. Unsere Sinnesorgane fungieren dabei als Eingabe-Einheiten; sie leiten die wahrgenommene Information an das Gehirn weiter: Das Auge entdeckt ein schönes Tier, die Ohren nehmen gute Musik auf und die feine Nase verrät einen Rinderbraten. Im Gehirn werden diese Eindrücke nicht nur registriert, sondern auch verarbeitet. Wehe, wenn sich in unserer Lieblingsmahlzeit tote Fliegen befänden – der Ruf nach dem bedauernswerten Ober bliebe bestimmt nicht aus. Hierbei würde sich dann unser Sprachorgan als wirkungsvolle Ausgabe-Einheit erweisen.
Und wenn man sich jetzt der Frage zuwendet, wie wohl der Zentral-Einheit die zu verarbeitenden Daten zugeführt werden, so braucht man sich beispielsweise nur an den Karten-Leser als Eingabe-Einheit zu erinnern.
In diesem Falle genügte ein einfacher Lesebefehl, um den Inhalt einer Lochkarte in den Karteneinlesebereich des Hauptspeichers gelangen zu lassen. Für jede andere Eingabe-Einheit müßte natürlich ebenfalls ein eigener Eingabebereich im Hauptspeicher definiert sein. Entsprechendes gilt auch für die Ausgabe-Einheiten: Ein Drucker benötigt für die Ausgabe von Daten aus dem Hauptspeicher einen Druckausgabebereich, ein Karten-Stanzer einen Stanzausgabebereich usw.
Damit wäre dann auch das letzte Glied in der Kette Dateneingabe – Datenverarbeitung – Datenausgabe erwähnt und der Weg für ausführlichere Betrachtungen von Ein/Ausgabe-Einheiten freigegeben.
Wie bereits bekannt, sind der Programmierung neben rein technischen und kaufmännischen Berechnungen auch organisatorische Verwaltungsarbeiten etc. zugänglich. Angefangen bei der automatischen Lohn- und Gehaltsabrechnung reichen diese über die Rech-

* Für Ein/Ausgabe-Einheiten hat sich auch der Name externe Einheiten oder Peripheriegeräte eingebürgert.

nungsschreibung, Fertigungssteuerung und Trendberechnung (Hochrechnungen bei Wahlen) bis zur Bahnkorrekturermittlung für gestartete Raketen.
Für mathematische Probleme, deren Lösung mitunter nur in einer einzigen auszudruckenden Zahl besteht, ist ein „schneller" Drucker von geringem Interesse. Hingegen wird man in einem Kaufhaus, in dem täglich Tausende von Rechnungen anfallen, größten Wert auf einen „schnellen" Drucker legen.
Die Ausstattung (Konfiguration) einer Datenverarbeitungsanlage mit Ein/Ausgabe-Einheiten wird sich also stets an dem zu bewältigenden Aufgabenkreis orientieren.
Bei größeren EDV-Systemen sind Konfigurationen mit zwei Druckern, einem Karten-Leser, einem Karten-Stanzer, sechs Platten-Einheiten und acht Band-Einheiten keine Seltenheit. Oft ist ein einziges Programm so umfangreich, daß es neben der Rechungsschreibung auch noch die Lagerhaltung, das Bestellwesen und Verbuchungen abwickelt (für jede der an einem Programm beteiligten Dateien muß ein eigener Ein- bzw. Ausgabe-Bereich im Hauptspeicher definiert werden).
Durch die phantastisch hohe Geschwindigkeit der Programmausführung kann unter anderem ein Computer der Betriebsleitung in kürzester Zeit wichtige Ergebnisse mitteilen: Umsatzzahlen, Gewinneinbußen, freie Maschinenkapazitäten usw. – der Betrieb bleibt dadurch weiterhin konkurrenzfähig, denn derartige Angaben haben oft den größten Wert für wirksame Entscheidungen. Eine leistungsfähige Wirtschaft ist deshalb ohne die elektronische Datenverarbeitung schon nicht mehr denkbar. Ständig bringt die Computerindustrie neue, leistungsfähigere Anlagen auf den Markt.
Für die verschiedensten Anwendungsgebiete sind besondere Ein/Ausgabe-Einheiten geschaffen worden; unter ihnen auch solche, die Handschriften maschinell lesen oder Ergebnisse akkustisch bekanntgeben können. Die Schwierigkeiten und Probleme, die die Techniker und Wissenschaftler bei der Entwicklung solcher Geräte zu bewältigen hatten, lassen sich nur erahnen. Schließlich kann zur Sprachausgabe kein Mensch in den Computer eingebaut werden, der über eine Lautsprecheranlage das Ergebnis ausruft. Die Worte müssen ausschließlich auf technischem Wege erzeugt werden. Wie das im einzelnen geschieht, ist höchst kompliziert, wäre aber auf einfache Art und Weise wie folgt zu erklären:
Angenommen, im Sprach-Ausgabebereich des Hauptspeichers stünde ein Kontenstand von 1 000 DM. Diese Zahl ist selbstverständlich binär verschlüsselt. Um sie als gesprochenes Wort „eintausend Mark" auszugeben, wäre man gezwungen, auf eine Art „Computer-Schallplatte" zurückzugreifen, auf der an bestimmten Stellen verschiedene Worte gespeichert sind. Für die Zahl 1 000 müßte gerade jene Stelle abgespielt werden, an der früher einmal das gesprochene Wort „eintausend" aufgenommen wurde. Für das „DM"-Zeichen wäre das Wort „D-Mark" abzuspielen. Das Aufsuchen der den einzelnen Worten zugeordneten Schallplattenstellen erfolgt programmgesteuert und völlig automatisch. Das Ergebnis ist verblüffend: Der Computer kann plötzlich reden. Bei entsprechender Programmierung ließe sich sogar erreichen, daß er für den Kontenstand 000 DM die munteren Worte „Sie sind pleite" oder „Bitte, kürzen Sie künftig das Haushaltsgeld" ausruft.
Manche Datenverarbeitungsanlagen können sogar auf einen telefonischen Anruf hin eine ganz bestimmte Auskunft erteilen. Sie wählen einfach eine Schlüsselzahl und Ihre Kontonummer. Die Schlüsselzahl soll beispielsweise den Wunsch ausdrücken, den Kontenstand zu erfahren. Der Computer hält für diesen Zweck ein bestimmtes Programm bereit. Sobald Ihre Kontonummer im Telefon-Einlesebereich des Hauptspeichers steht wird über dieses Programm jene Stelle auf der Magnetplatte ermittelt, wo Ihr Kontenstand gespeichert ist. Alles weitere läßt sich bereits ohne Schwierigkeiten erahnen: Der Kontenstand

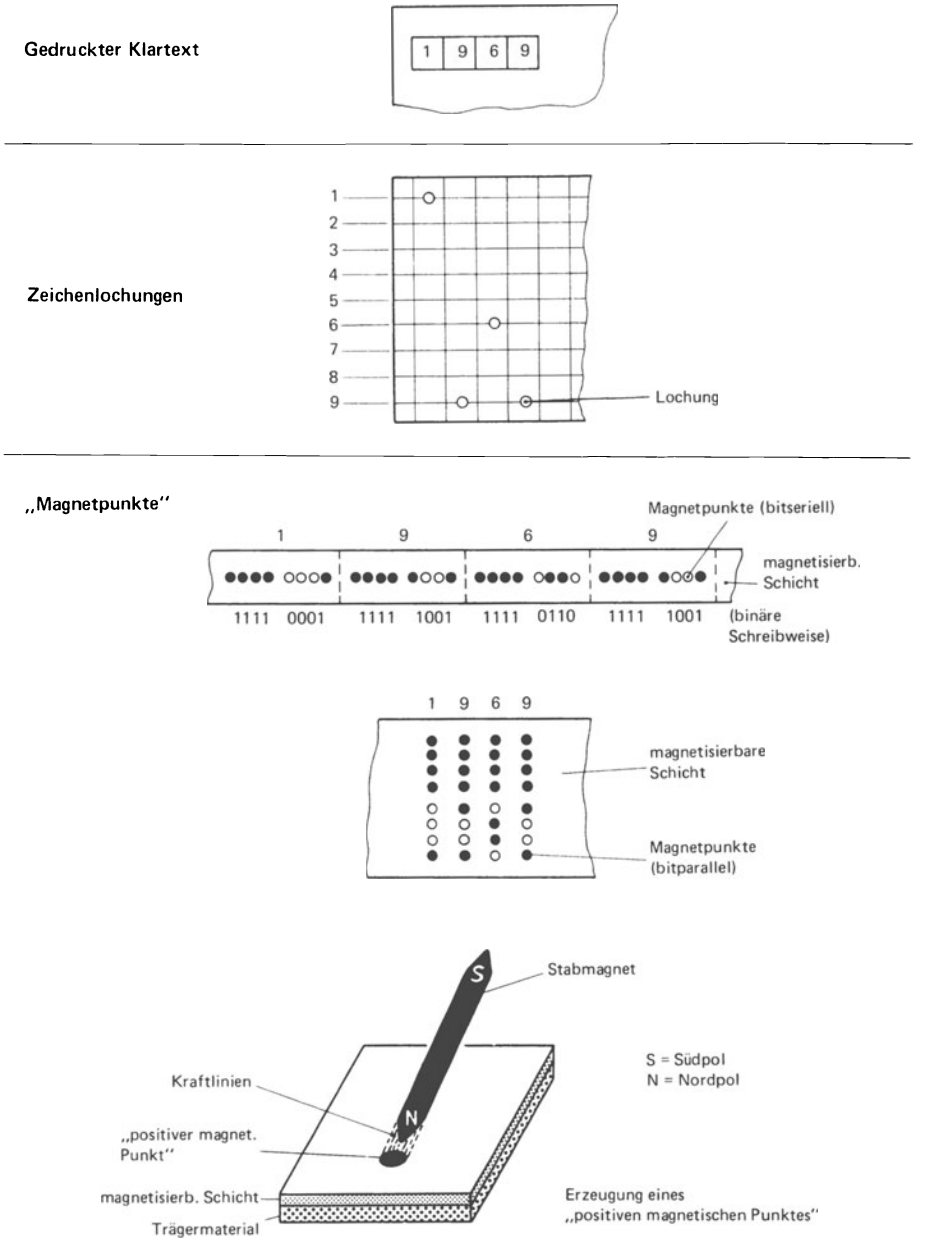
wird in den Hauptspeicher eingelesen, in den Telefon-Ausgabebereich übertragen und sodann als gesprochenes Wort telefonisch übermittelt.
Möglich, daß auch der Tag nicht mehr fern ist, an dem ein Computer auf ein gesprochenes Wort reagieren kann. Wie das nun wieder möglich wäre? Ganz einfach, die über ein Mikrophon aufgenommenen Sprachschwingungen werden elektronisch analysiert und die ihnen zukommende Bedeutung in Form von Zahlen und Buchstaben ermittelt. Damit könnte dann aber der Computer in bereits gewohnter Weise umgehen.
Damit erkennen wir, wie wichtig in der Computertechnik die maschinelle Datenein- und Datenausgabe ist. Es werden Peripherie-Geräte benötigt, die nach technisch gut zu realisierenden Prinzipien arbeiten. Dieser Forderung genügen insbesondere die Möglichkeiten, Daten

als Zeichenlochungen,
als „magnetische Punkte"
oder in visuell erkennbarer Form

darzustellen.
Bei der zweiten Methode macht man sich die physikalische Tatsache zunutze, daß mit Hilfe eines (stabförmigen) Magneten auf einer magnetisierbaren Schicht kleine magnetische Bereiche („Punkte") erzeugt werden können. Dabei ist der Magnet ähnlich einem Bleistift zu handhaben; ohne eines seiner beiden „Enden" (Nord- bzw. Südpol) direkt auf der magnetisierbaren Schicht aufsetzen zu müssen, wird bereits bei einem geringen Abstand durch die vom „Magnetpol" ausgehenden Kraftlinien ein „magnetischer Punkt" erzeugt (Abb. 128):

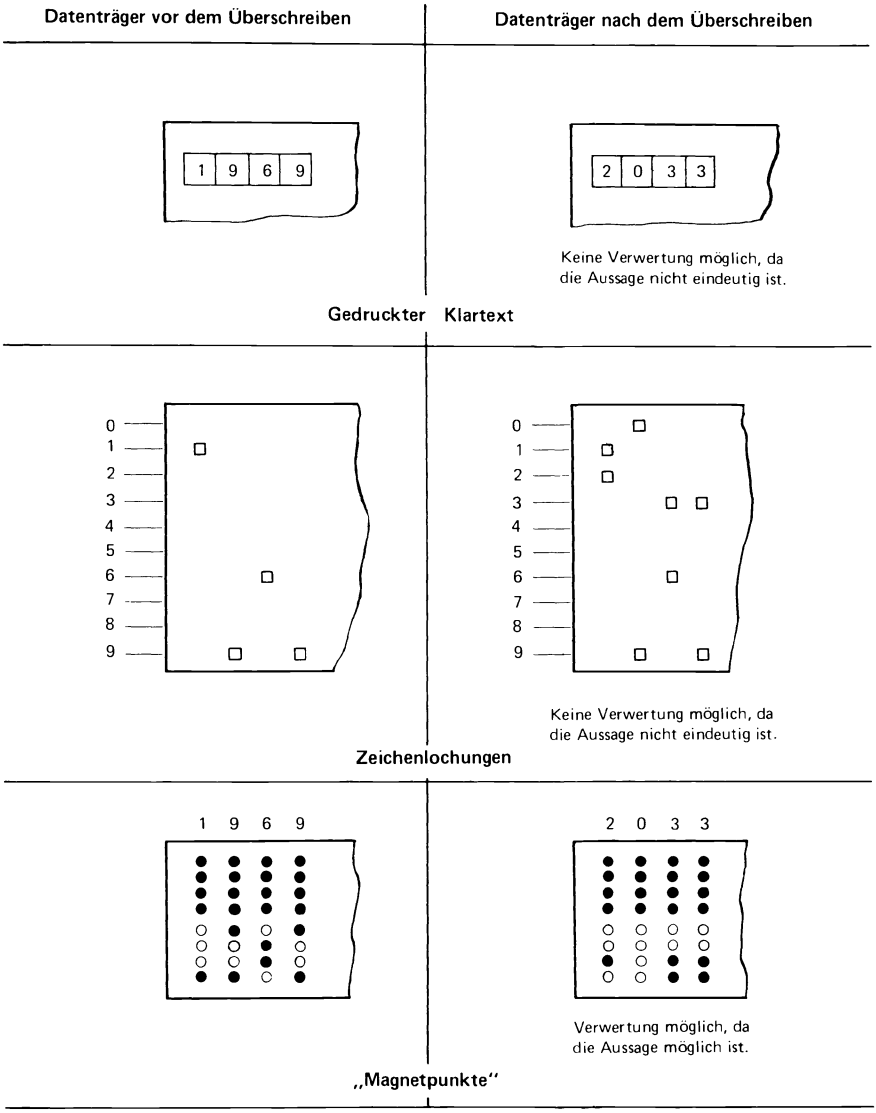
Durch Einwirkung des Nordpoles entsteht ein „positiver magnetischer Punkt ● (in binärer Darstellung als „1" gekennzeichnet);
Durch den Südpol hingegen wird ein „negativer magnetischer Punkt" o gebildet (in binärer Darstellung als „0" gekennzeichnet).

Während ein voll beschriebenes oder mit Zeichenlochungen versehenes Blatt Papier (Abb. 129) zur Aufnahme neuer Daten völlig ungeeignet ist (wer schreibt schon gern übereinander), können „magnetische Punkte" überschrieben, d. h. durch neue „Magnetpunkte" ersetzt werden (Abb. 129). Dabei wird durch erneutes Einwirken des Nord- oder Südpoles des Magneten ein neuer magnetischer Punkt erzeugt, unabhängig davon, ob diese Stelle vorher „positiv" oder „negativ magnetisch" war. Eine magnetisierbare Schicht ist somit immer wieder verwendbar. (Denken Sie bitte an das Überspielen eines Tonbandes).
Die peripheren Einheiten können durch einen entsprechenden Programmbefehl zu einer selbstätigen Datenein- bzw. Datenausgabe veranlaßt werden.
Sie müssen also auch über einen eigenen Ein- bzw. Ausgabebereich verfügen. Die Zuordnung eines solchen Bereiches zu einer bestimmten externen Einheit erfolgt im Programm. Dort wird der Name des E/A-Bereiches (E=Eingabe; A=Ausgabe) und seine Größe festgelegt.
Es soll nicht Aufgabe dieses Buches sein, sämtliche Ein/Ausgabe-Einheiten erschöpfend zu behandeln. Aber einige Hinweise auf die bekanntesten von ihnen erscheinen doch angebracht, weil sie das Bild von komplexen Computersystemen abrunden helfen:



Verschiedene Möglichkeiten der Datendarstellung (am Beispiel der Zahl 1969)

Abb. 128



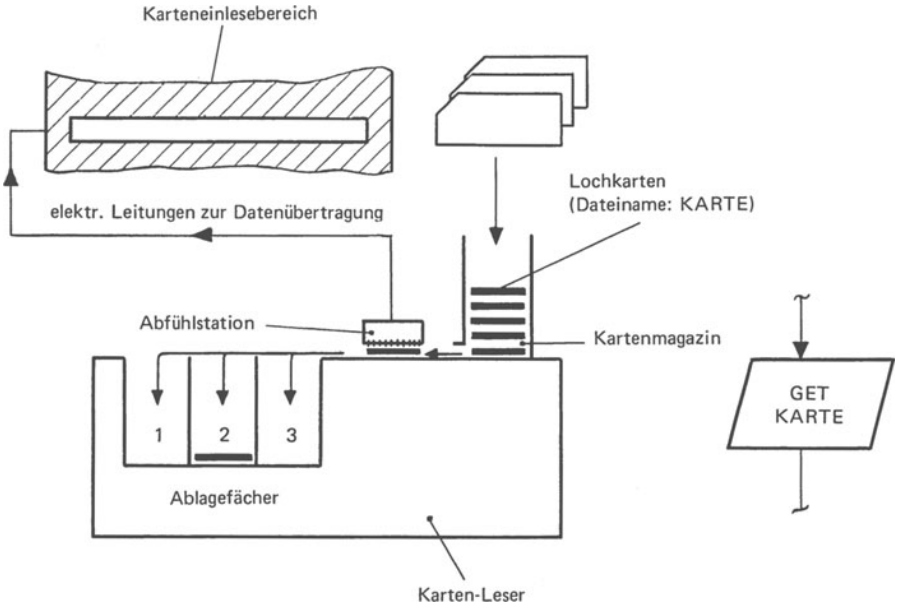
Überschreiben alter Daten (1969) durch neue Daten (2033)

Abb. 129

A. Peripherie-Einheiten für Aufzeichnungsträger mit Zeichenlochungen

I. Der Kartenleser

Er wurde bereits ausführlich in dem Kapitel 3 behandelt. Darum sei hier in Abb. 130 nur noch einmal kurz sein Prinzip wiedergegeben:



Schema eines Karten-Lesers bei der Auswirkung eines Lesebefehles

Abb. 130

Durch den Lesebefehl GET KARTE werden folgende Vorgänge ausgelöst:

1. Automatischer Transport der im Kartenmagazin zuunterst liegenden Lochkarte in die Abfühlstation.
2. Übertragung des Inhaltes dieser abgefühlten Lochkarte in den Karteneinlesebereich des Hauptspeichers.
3. Ablage dieser Lochkarte im Fach 2, falls kein besonderer Programmbefehl ihre Ablage in einem anderen Fach vorsieht.

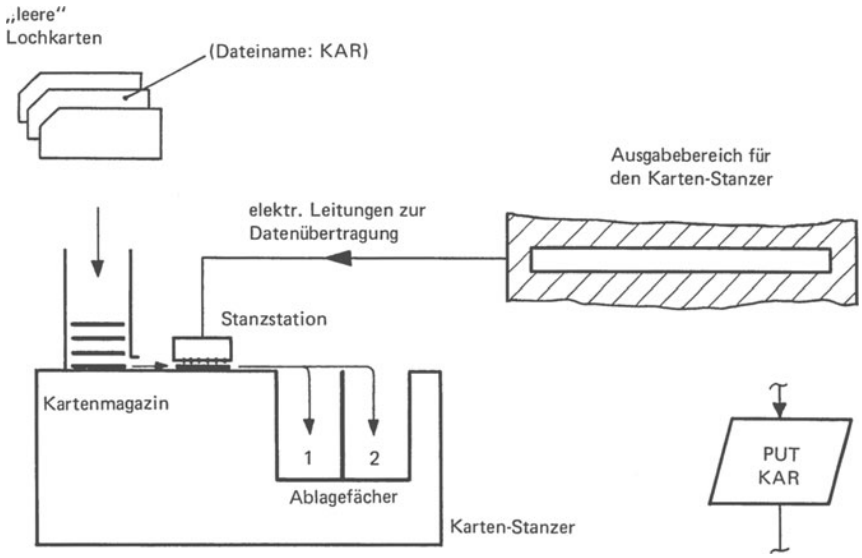
II. Der Kartenstanzer

Er ermöglicht, wie in Abb. 131 dargestellt, die automatische Ausgabe von Daten in Lochkarten.

Durch den Ausgabebefehl (Stanzbefehl) PUT KAR werden folgende Vorgänge ausgelöst:

1. Automatischer Transport der im Kartenmagazin zuunterst liegenden „leeren" Lochkarte in die Stanzstation.

2. Der Inhalt des Ausgabebereiches für den Karten-Stanzer wird in Form von Zeichenlochungen in diese Lochkarte gestanzt.
3. Ablage dieser Lochkarte im Fach 1, falls kein besonderer Programmbefehl ihre Ablage in dem anderen Fach vorsieht.

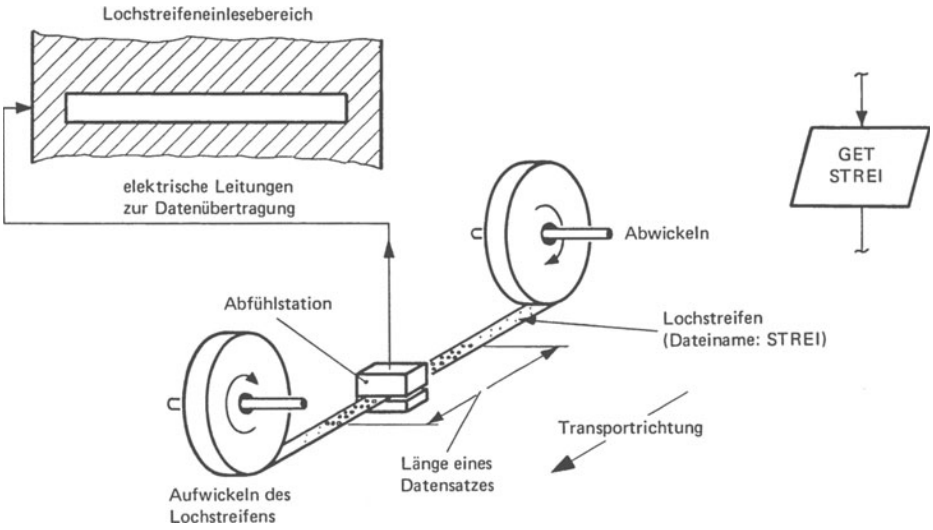


Schema eines Karten-Stanzers bei der Auswirkung eines Ausgabebefehles

Abb. 131

III. Der Lochstreifen-Leser

Er ermöglicht die Übertragung der in einem Lochstreifen (Abb. 132 und Abb. 133) abgelochten Zeichen in den Hauptspeicher; ein Vorgang, der sich ähnlich dem Einlesen einer Lochkarte vollzieht. Die Übertragung jeweils in der Länge eines Datensatzes, der durch entsprechende Lochungen auf dem Lochstreifen abgegrenzt ist. (In Abb. 134 ist der 8-Kanal-Lochstreifencode wiedergegeben).
Durch den Lesebefehl GET STREI werden folgende Vorgänge ausgelöst:
1. Der Lochstreifen wird in der Länge eines Datensatzes zeichenweise unter der Abfühlstation hindurchbewegt.
2. Übertragung des Inhaltes des abgefühlten Lochstreifensatzes in den Lochstreifeneinlesebereich des Hauptspeichers.



Schema eines Lochstreifen-Lesers bei der Auswirkung eines Lesebefehles

Abb. 132

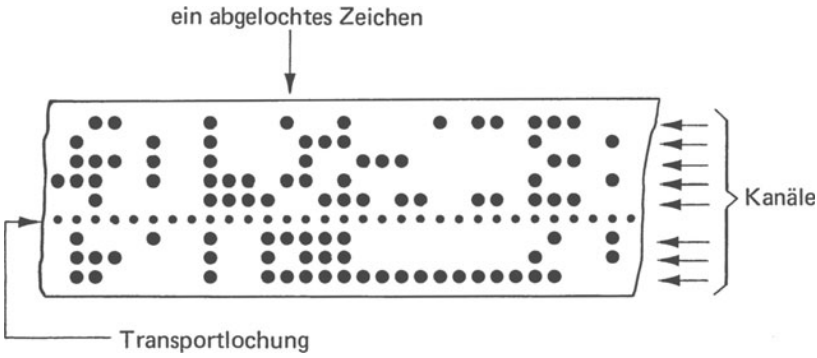


Abb. 133

Zeichen	Lochstreifen-code
A	1000 0001
B	1000 0010
C	1100 0011
D	1000 0100
E	1100 0101
F	1100 0110
G	1000 0111
H	1000 1000
I	1100 1001
J	1101 0001
K	1101 0010
L	1001 0011
M	1101 0100
N	1001 0101
O	1001 0110
P	1101 0111
Q	1101 1000
R	1001 1001
S	1110 0010
T	1010 0011
U	1110 0100
V	1010 0101
W	1010 0110
X	1110 0111
Y	1110 1000
Z	1010 1001
0 Null	1111 0000
1	1011 0001
2	1011 0010
3	1111 0011
4	1011 0100
5	1111 0101
6	1111 0110
7	1011 0111
8	1011 1000
9	1111 1001

Abb. 134

IV. Der Lochstreifen-Stanzer

Diese Einheit ermöglicht die direkte Ausgabe von Daten aus dem Hauptspeicher in den Lochstreifen. Von dieser Möglichkeit wird vielfach auf dem Gebiet der Datenfernübertragung Gebrauch gemacht. Dabei ist der Computer über Fernleitungen mit dem Lochstreifen-Stanzer verbunden.

B. Peripherie-Einheiten für magnetische Aufzeichnungsträger

Bei ihnen wird im Falle der Datenaufzeichnung (Schreiben) bzw. des Datenlesens (Lesen) eine magnetisierbare Schicht unter einem Schreib-Lese-Kopf (S-L-Kopf) hindurchbewegt (Abb. 135, 136, 137).

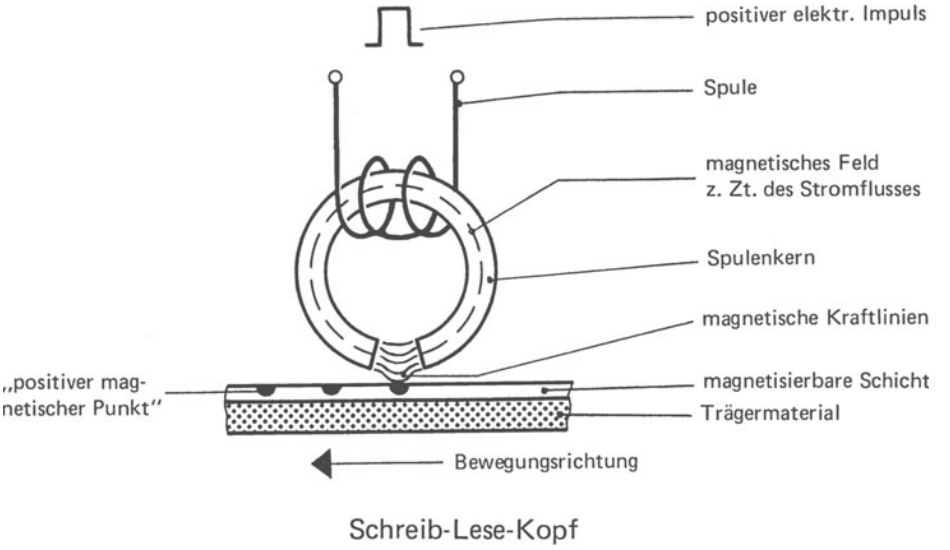


Abb. 135

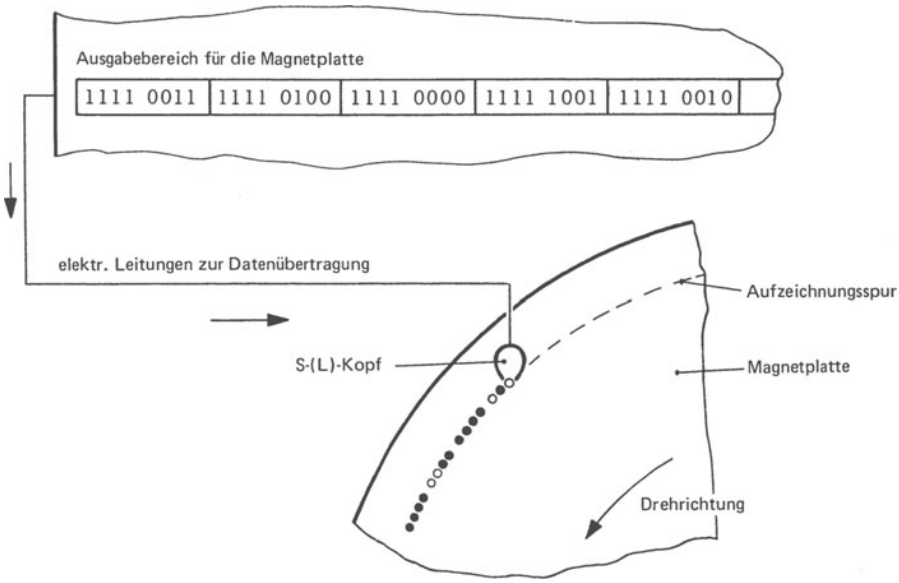
Der S-L-Kopf kann dabei, je nach Anforderung, eine von insgesamt zwei möglichen Funktionen ausführen: Schreiben oder Lesen.

Schreibvorgang

Soll Information aufgezeichnet werden, so erzeugt er auf der sich vorbeibewegenden magnetisierbaren Schicht „positive oder negative magnetische Punkte".
Kurzzeitige elektrische Impulse (positive oder negative), die von der Zentraleinheit des Computers geliefert werden, bewirken in der Spule des S-L-Kopfes einen Stromfluß, durch den wiederum im Spulenkern kurzzeitig ein Magnetfeld entsteht (Abb. 135). Die auf die magnetisierbare Schicht einwirkenden magnetischen Kraftlinien erzeugen einen „magnetischen Punkt". Seine Polarität (positiv oder negativ) richtet sich danach, ob die Spule durch einen positiven oder negativen elektrischen Impuls erregt wurde.

I. Magnetplatten- und Magnetband-Einheiten

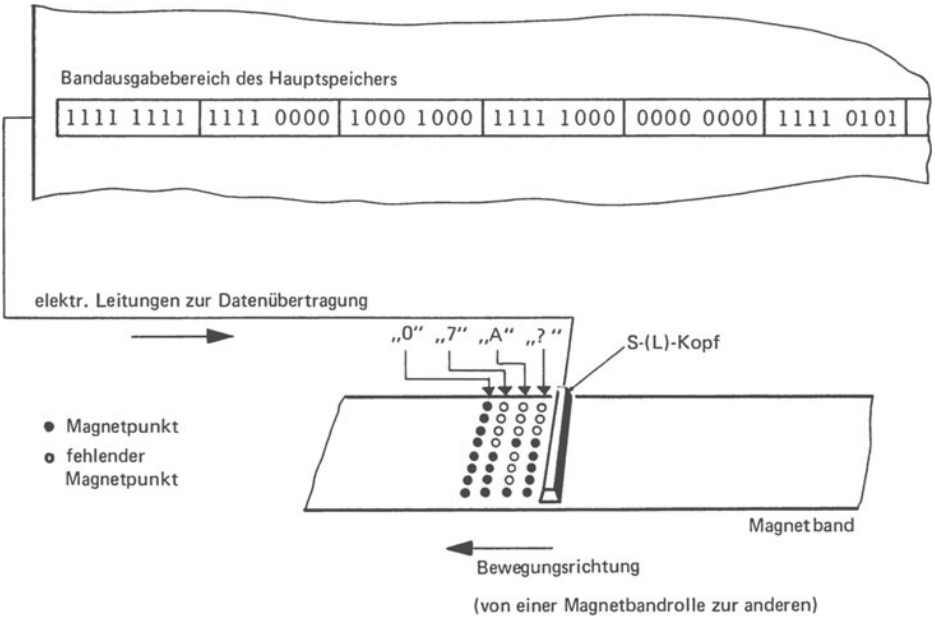
Wegen ihrer großen Verbreitung und Bedeutung sind ihnen eigene Kapitel gewidmet. Deshalb sei hier nur einleitend ihre prinzipielle Wirkungsweise erläutert.
Während der S-L-Kopf einer Magnetplatte die „magnetischen Punkte" („Datenbits") nacheinander (bitseriell) erzeugt (Abb. 136), kann sie der S-L-Kopf eines Magnetbandes bitparallel ausgeben (Abb. 137).



Datenaufzeichnung auf der Magnetplatte

Abb. 136

Die aufzuzeichnende Information steht immer im entsprechenden Aufgabebereich des Hauptspeichers. Die Zeichen sind dort üblicherweise durch ein- und ausgeschaltete Speicherzellen dargestellt (Abb. 136, 137). Sobald ein Schreibbefehl erteilt wird, werden diesen Speicherzellen entsprechend positive bzw. negative elektrische Impulse erzeugt und an den S-L-Kopf weitergeleitet. Dieser setzt sie in „magnetische Punkte" (Datenbits) um, so daß schließlich der Inhalt des Ausgabebereiches auf dem magnetischen Aufzeichnungsträger erscheint.



Datenaufzeichnung auf dem Magnetband

Abb. 137

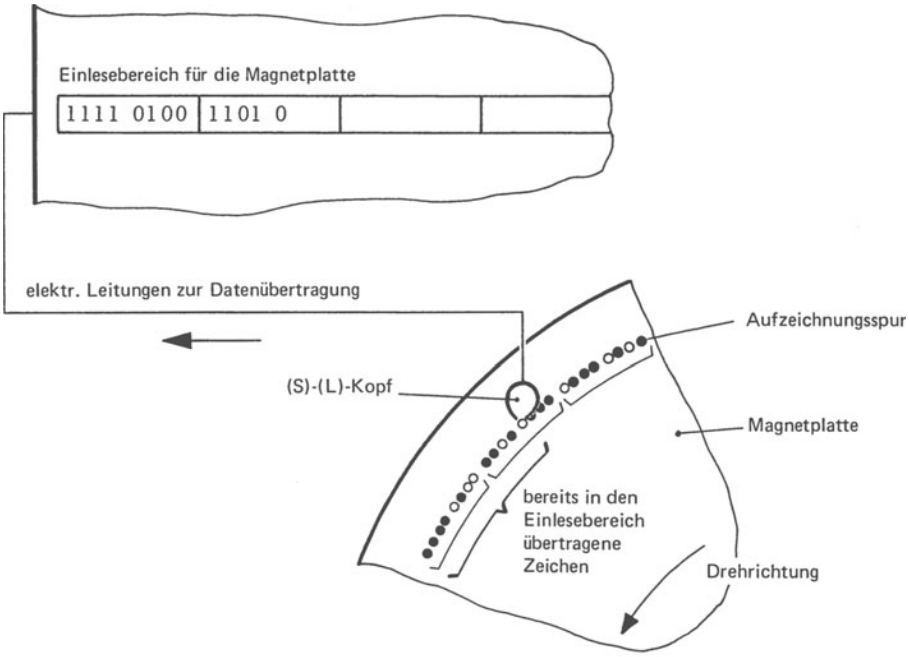
Lesevorgang

Der Name (Schreib)-Lese-Kopf weist darauf hin, daß er neben der bereits erwähnten Datenaufzeichnung auch ein Lesen von „Magnetpunkt"-Information gestattet. Dieser Vorgang wird durch einen Lesebefehl eingeleitet. Sobald ein „magnetischer Punkt" (positiv oder negativ) unter dem S-L-Kopf hindurchbewegt wird, entsteht an dessen Spulenden ein elektrischer Impuls (positiv bzw. negativ). Dieser Impuls schaltet im Einlesebereich des Hauptspeichers die entsprechende Speicherzelle ein oder aus (Abb. 138). Nach der Ausführung eines Lesebefehls steht der Inhalt des auf dem magnetischen Aufzeichnungsträger gespeicherten Datensatz im Einlesebereich des Hauptspeichers und ist somit einer Programmverarbeitung zugänglich.
Es ist also möglich, den S-L-Kopf sowohl für eine Schreib- als auch für eine Leseoperation zu benutzen.

II. Der Trommelspeicher

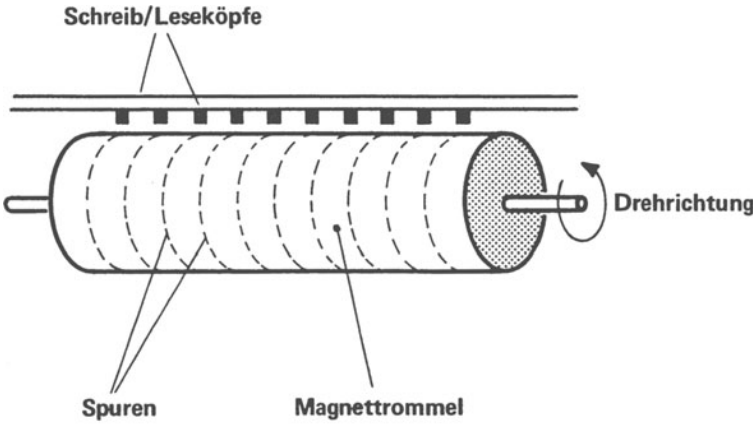
Der Trommelspeicher ist mit der Magnetplatte verwandt. Er enthält eine rotierende Trommel, deren Oberfläche magnetisierbar ist und der Zeichenspeicherung dient. Jeder

Spur ist ein Schreib-Lese-Kopf zugeordnet (Abb. 139, der unmittelbar über der Trommeloberfläche angebracht ist. Ein solcher Trommelspeicher wird als zusätzlicher Speicher eingesetzt, wenn ein schneller, direkter Zugriff zu den einzelnen Datensätzen erforderlich ist.



Datenlesen von der Magnetplatte

Abb. 138

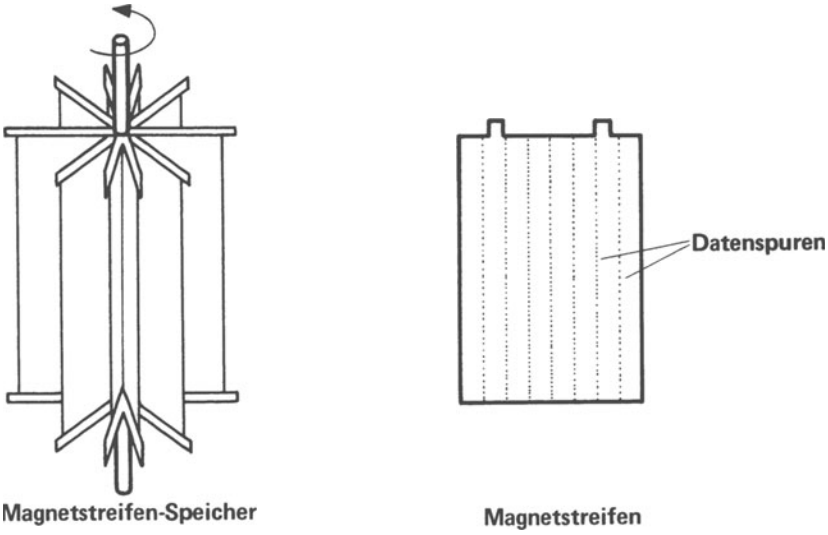


Prinzip des Magnettrommelspeichers

Abb. 139

III. Der Magnetstreifen-Speicher

Er ist im wahrsten Sinne des Wortes ein Riese, denn sein Speichervermögen ist fast unbegrenzt (400 Millionen Zeichen). Die Zeichen sind auf Magnetstreifen (Abb. 140) gespeichert, die lamellenförmig in einem drehbaren Ständer aufgehängt sind.
Bei Bedarf wird einer dieser Streifen automatisch in Bruchteilen einer Sekunde ergriffen, unter Vakuum auf eine Trommel gespannt und schon können nach dem Trommelspeicher-Prinzip Zeichen gelesen oder geschrieben werden.



Prinzip des Magnetstreifenspeichers

Abb. 140

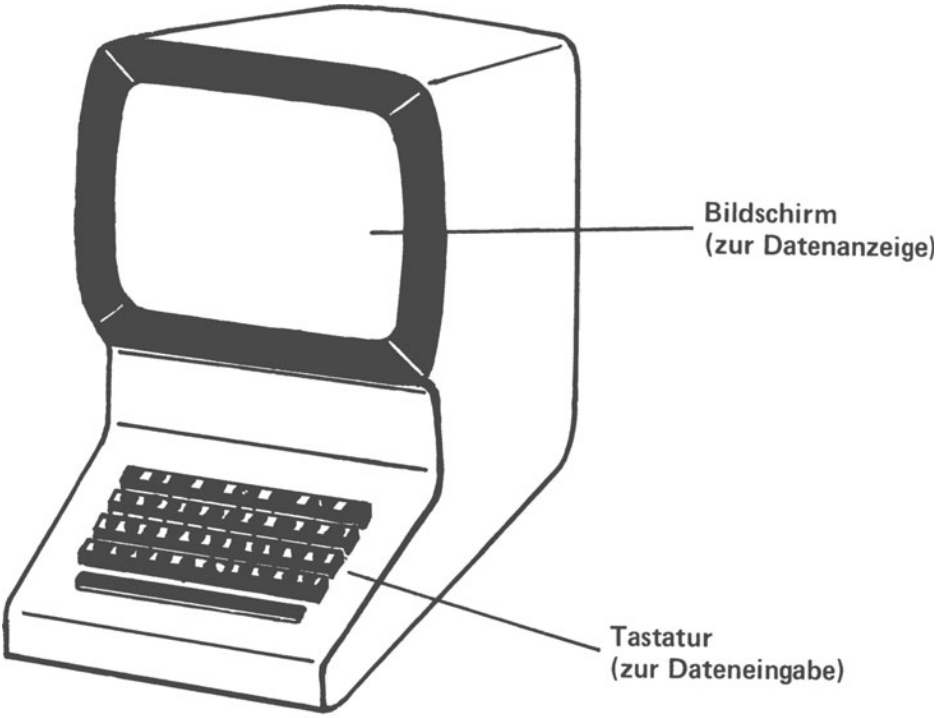
C. Peripherie-Einheiten für Daten in visuell erkennbarer Form

Eine solche Klassifizierung ist sehr umfassend. Wir wollen dabei in erster Linie an Drucker und Bildschirmgeräte denken.
Dem Drucker ist wegen seiner klassischen Bedeutung als Datenausgabe-Einheit ein eigenes Kapitel (19) gewidmet.
Bildschirmgeräte erinnern uns an das Fernsehen. Warum sollte es nun nicht möglich sein, einen Bildschirm an einen Computer anzuschließen, auch wenn dieser nur nüchterne Zahlen und Buchstaben liefert.
Dabei müßte lediglich dafür gesorgt werden, daß die im Bildschirmausgabebereich stehende codierte Information (beispielsweise die Bitkombination 1100 0101 für das Zeichen E, um nur eines zu nennen) durch eine automatisch wirkende elektronische Schaltung in ein als E zu erkennendes Zeichen umgewandelt wird.

Aufbauend auf dieser Anregung könnte man jetzt auch auf die Idee kommen, Lampenfelder zur Anzeige von Computerdaten zu verwenden. Die Ausführung wäre denkbar einfach. Es müßte nur eine Schaltung entwickelt werden, die aus einem zunächst noch codierten Zeichen diejenigen Lampen des Feldes einschaltet, welche zur Darstellung erforderlich sind.
Mit dieser Vorstellung des Lampenfeldes können wir uns auch – zumindest in vereinfachender Form – die Wirkungsweise moderner Plasmaanzeigetafeln erklären. Diese Anzeigetafeln bestehen aus einer Vielzahl winziger Lichtzellen, welche an- bzw. ausgeschaltet werden können.

I. Bildschirmgeräte (Datenstationen)

Der Einsatz solcher Geräte (Abb. 141) als Ein/Ausgabe-Einheiten hat innerhalb der letzten Jahre explosionsartig zugenommen.
Der Datenausgabe dient der Bildschirm, zumeist der einer Kathodenstrahlröhre. Auf ihm erscheinen die Daten in deutlich lesbarer Form, z. B. in 16 Reihen a 64 Zeichen.



Datensicht-Station

Abb. 141

Zur Dateneingabe ist die Datenstation mit einer Tastatur versehen. Soll nun ein bestimmtes Zeichen in eine bestimmte Bildschirmposition eingegeben werden, so ist lediglich zweierlei zu tun:
Die gewünschte Position ist durch ein durch Tastendruck in alle Stellen des Bildschirmes zu bewegendes Markierungszeichen zu kennzeichnen und dann ist nur noch die entsprechende Zeichentaste zu betätigen. Unmittelbar darauf erscheint an der vormals markierten Stelle das gewünschte Zeichen, und nicht genug damit, es steht auch in codierter Form im Bildschirmspeicher und ist so einer weiteren Programmverarbeitung zugänglich.
Der praktische Nutzen der Bildschirmgeräte ist äußerst vielfältig. So kann sich z. B. der kaufmännische Direktor einer Firma auf einen einzigen Knopfdruck hin den Tagesumsatz zur Anzeige bringen lassen. Selbstverständlich muß dieser Umstand im Programm entsprechend berücksichtigt worden sein (Abb. 142).

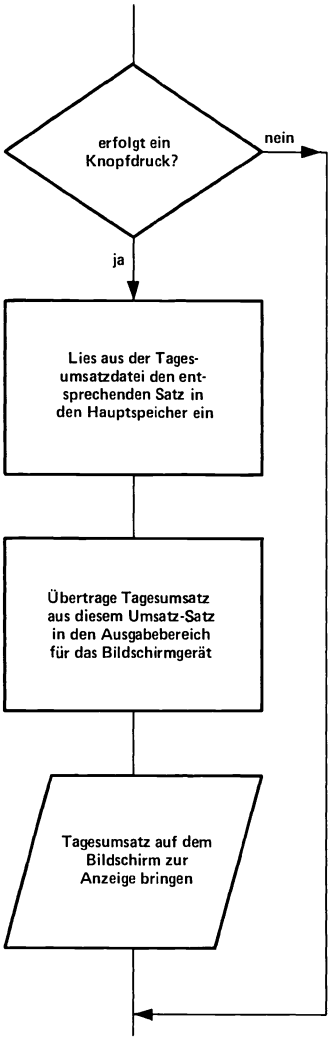


Abb. 142

II. Graphische Ausgabe-Einheiten

An bestimmte Computermodelle lassen sich sogenannte Plotter anschließen, die die errechneten Ergebnisse in Form von Kurven auf Papier aufzeichnen. Dabei wird ein Schreibgerät so über einer zu beschriftenden Unterlage geführt, wie es bei vielen Registrierschreibern der Fall ist.
Außerdem ist es aber auch möglich, Rechenergebnisse in Form von Diagrammen oder Kurven auf Bildschirmgeräten darzustellen.

III. Belegleser

Mit den Fortschritten auf dem Gebiet der Zeichenerkennung gewinnen die Belegleser immer mehr an Bedeutung. So ist es heute bereits möglich, nicht nur Strichmarkierungen und bestimmte Maschinenschriften, sondern auch Handschriften (Blockschrift) zu erfassen. Für diese unterschiedlichen Möglichkeiten des Beleglesens wurden verschiedene Belegleser entwickelt.

a) Markierungsleser

Sie ermöglichen die Feststellung, an welchen Stellen eines Formulars dicke Bleistiftstriche als Markierungen angebracht sind. Jeder dieser Stellen ist per Programm ein bestimmter Wert zugeordnet. Abb. 143 zeigt ausschnittsweise ein Formular, auf dem die Liefernummer 1488 markiert ist. Nach dem Abfühlen der markierten Stellen erscheint dann im Markierungsleser-Einlesebereich des Hauptspeichers die Zahl 1488.

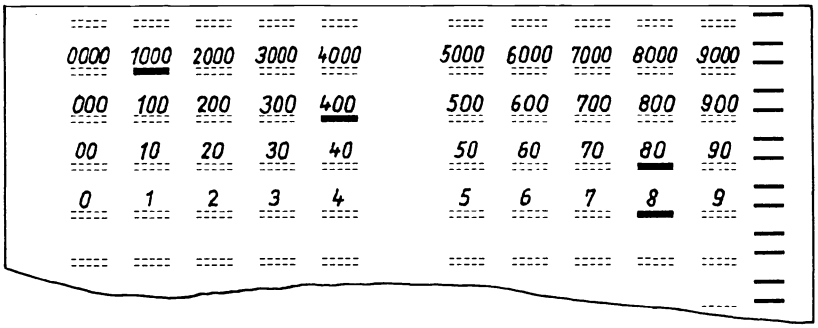


Abb. 143

b) Klarschrift-Leser

Unter den Klarschrift-Lesern gibt es Konstruktionen, die entweder nur Ziffern oder Ziffern und Buchstaben einer bestimmten stilisierten Maschinenschrift (Abb. 144) oder Handschrift (Blockschrift) lesen können.
In den sogenannten Mehrfunktions-Beleglesern sind alle diese Möglichkeiten berücksichtigt. Als Belege können Formulare unterschiedlicher Größe und Aufteilung, sowie Endlosstreifen von Registrierkassen und Additionsmaschinen ausgewertet werden.

502424525242	C
202501271	X
15100000	N
11012001	S
8012345080	Z
2557115	Z
15100000	N
22012001	S
807456512	Z
83457432	Z
93608908	Z
154325678	Z
1620756	Z
19876321	Z

Abb. 144

c) Magnetschrift-Leser

Kennzeichnend für derartige Einheiten ist die Tatsache, daß sie nur solche Zeichen lesen können, die mit einer eisenoxydhaltigen Tinte in einer ganz bestimmten Schriftart gedruckt worden sind. Beim Lesevorgang ist das Eisenoxyd aufmagnetisiert, so daß sich der Verlauf des Schriftzuges aufgrund des Magnetfeldes abtasten läßt.

Dieser kurze Streifzug in das Gebiet spezieller Ein/Ausgabe-Einheiten verrät, wie komplex das gesamte Gebiet der elektronischen Datenverarbeitung ist.

Sie bemerken, daß das ganze Gebiet der elektronischen Datenverarbeitung immer umfangreicher wird – der Weg, den Sie im Rahmen dieser Einführung bisher beschritten haben, gabelt sich immer mehr auf. Der Slogan

In allen Lebenslagen den Computer fragen

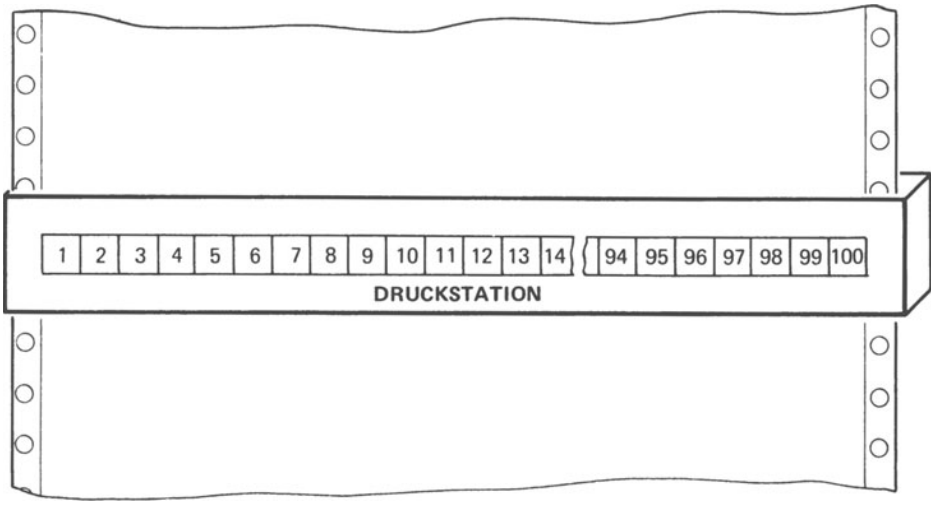
gewinnt immer mehr an Raum: Eine automatisierte Heirats- und Wohnungsvermittlung ist durchaus schon eine Realität. Englische Mediziner warten mit einer elektronischen Voraussage eines Herzinfarktes auf. Amerika kennt die Arbeitsvermittlung durch den Computer. Verbrechen und das Wetter werden im voraus errechnet und vieles andere mehr.

19. Der Drucker

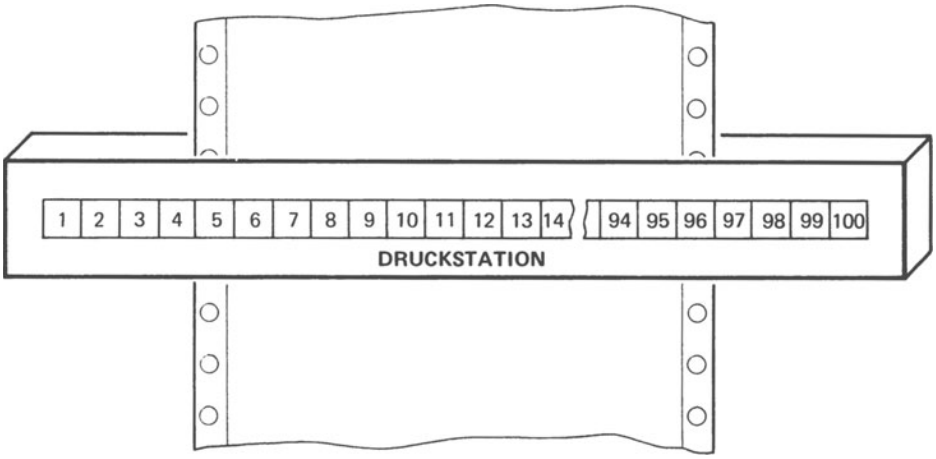
Der Drucker ist eine Ausgabe-Einheit, die wohl das augenscheinlichste Ergebnis eines Programmlaufes liefert, nämlich Listen. Das können entweder Rechnungen, Lieferscheine, statistische Aufstellungen, Kontoauszüge oder Gehaltszettel etc. sein.
Aus den vorangegangenen Kapiteln ist bereits bekannt, daß ein Formular immer zeilenweise bedruckt wird. Durch den Befehl

```
PUT      DRUCK
```

gelangt der gesamte Inhalt des Druckausgabebereiches des Hauptspeichers als Klarschriftzeile auf das Papier. Anschließend wird automatisch das Formular um eine Zeile unter der Druckstation weiterbewegt (1-Zeilen-Vorschub), so daß nach erneuten Druckbefehlen alle Zeilen untereinander erscheinen (Abb. 145).
Je nach dem Typ des Druckers kann eine Zeile bis zu 100, 132 oder gar noch mehr Druckpositionen umfassen. Die Formularbahnen dürfen je nach Erfordernis eine unterschiedliche Breite aufweisen.
Bei Mitarbeiterlisten (Abb. 26) würde man sich beispielsweise mit verhältnismäßig schmalen Formularen begnügen, bei umfangreichen statistischen Aufstellungen hingegen nicht. Danach werden einmal weniger, das andere Mal mehr Druckpositionen ausgenutzt. Der Druckausgabebereich muß jedoch immer genau so viele Stellen umfassen, wie es Druckpositionen gibt. Für jede Druckposition, in der nichts ausgedruckt werden soll, ist in der entsprechenden Stelle des Druckausgabebereiches stets ein Blank vorzusehen (Abb. 145).
Die Frage, wie die Zeichen gedruckt werden, ist zweifelsohne nicht nur für den Techniker von Interesse. Oftmals wird dabei von einer sogenannten Druckkette Gebrauch gemacht (Abb. 146). Die einzelnen Glieder dieser Kette bestehen aus den Typen für Ziffern, Buchstaben und Sonderzeichen. Die mit hoher Geschwindigkeit angetriebene Kette läuft an vielen, dicht nebeneinander liegenden Hämmerchen vorbei. Sobald eines der Hämmerchen (auf elektromagnetischem Wege angeregt) blitzartig Formular und Farbtuch gegen die sich gerade vorbeibewegende Drucktype schlägt, gelangt das betreffende Zeichen auf dem Papier zum Abdruck.
Eine raffiniert ausgedachte Konstruktion sorgt dafür, daß jedes Zeichen nur in der gewünschten Position gedruckt wird. So erscheint die Druckzeile schließlich als „Abbild" des Druckausgabebereiches auf der Liste.
Mit Hilfe der Druckkette lassen sich Leistungen bis zu 2000 Zeilen pro Minute erreichen.
Der Formularvorschub erfolgt mittels Stachelräder (Abb. 147). Diese drehen sich ruckweise um ihre Achse — ihre Zähne greifen dabei in die Transportlochungen der Formularbahn, die schrittweise an der Druckstation vorbeibewegt wird.
Als Beispiel sei eine Rechnungsschreibung erwähnt.
Je nach der Menge der Auftrags-Karten ist durch das Programm eine entsprechende Anzahl von Rechnungen zu erstellen.
Dabei erweist es sich als sehr vorteilhaft, vorgedruckte Endlosformulare (Abb. 148) zu verwenden. Die Aufteilung des Rechnungskopfes, der Artikelzeilen etc. ist fest vorgegeben und vom Programmierer genau zu beachten.
Wäre man nur auf den 1-Zeilen-Transport angewiesen, so müßte für das Bedrucken eines Rechnungsformulares folgender Weg beschritten werden (siehe Abb. 148):

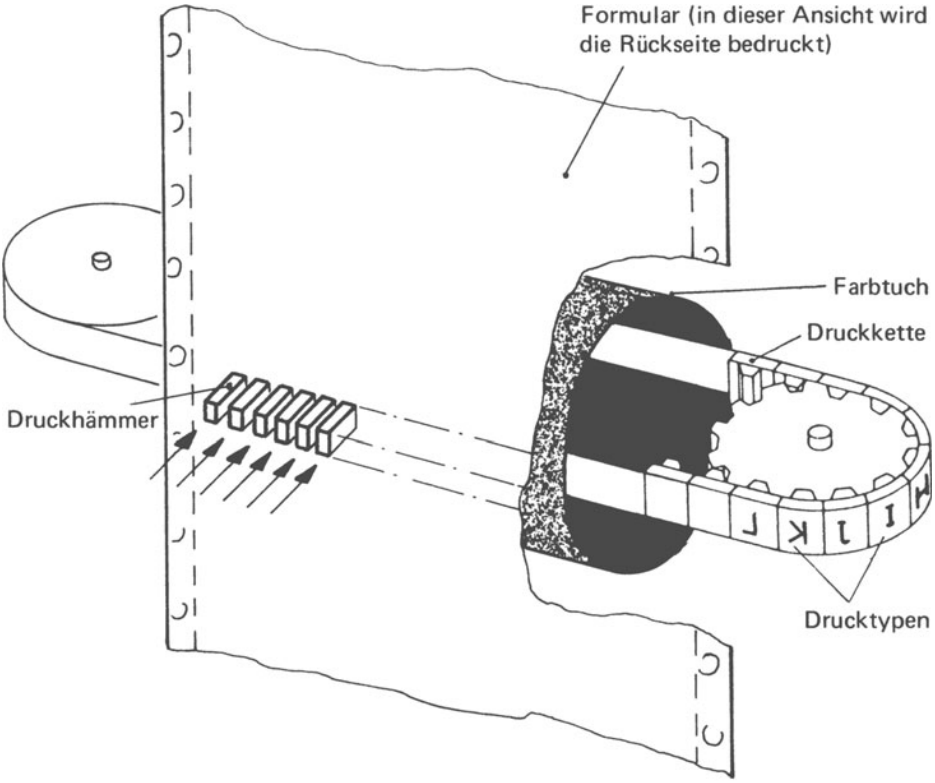


breites Formular (alle Druckpositionen werden benutzt)



schmales Formular (es werden nur die Druckpositionen 6–95 benutzt).

Abb. 145



Prinzip eines Kettendruckers

Abb. 146

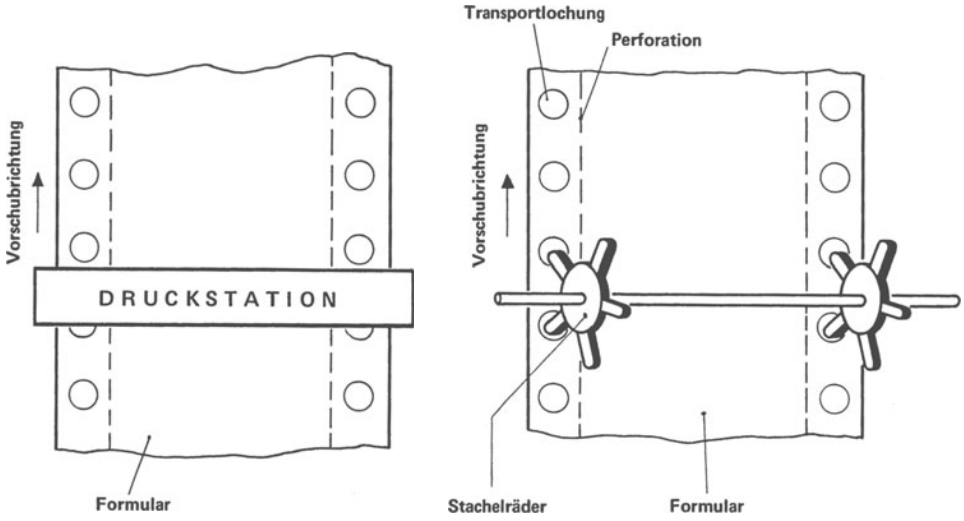


Abb. 147

Leerzeile drucken
Leerzeile drucken
Leerzeile drucken
Leerzeile drucken
Leerzeile drucken
Leerzeile drucken
Leerzeile drucken
Datumszeile drucken
Leerzeile drucken
Leerzeile drucken
Kundenanschrift (Teil A) drucken
Kundenanschrift (Teil B) drucken
Kundenanschrift (Teil C) drucken
Leerzeile drucken
Leerzeile drucken
Leerzeile drucken
Leerzeile drucken
1. Artikelzeile drucken
2. Artikelzeile drucken
3. Artikelzeile drucken
usw.

(Vor jedem Druckbefehl ist in den Druckausgabebereich die gewünschte Information zu übertragen. Leerzeilen erfordern Blanks).
Es wäre mehr als müßig, zur Überbrückung größerer Zwischenräume auf dem Formular häufig Leerzeilen ausgeben zu müssen.

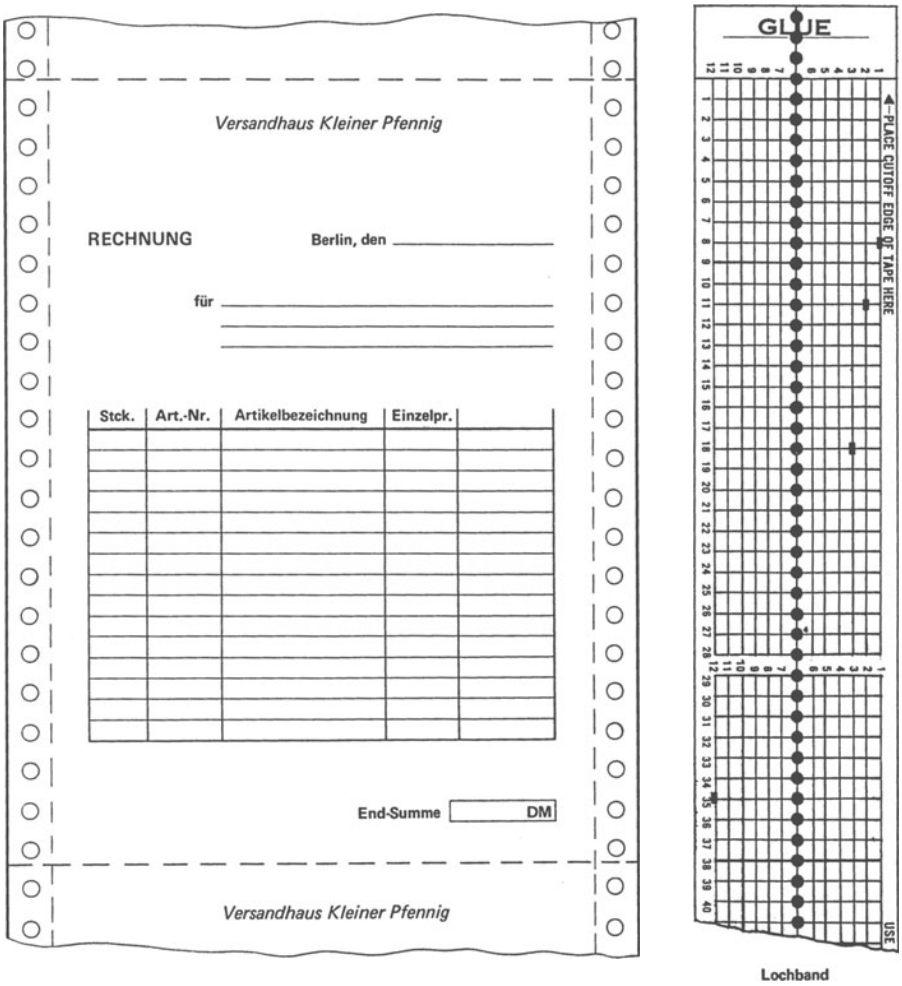


Abb. 148

Aus diesem Grunde benutzt man ein Lochband zur Vorschubsteuerung. Es ermöglicht einen Formularvorschub über mehrere Zeilen hinweg. Je nach den Erfordernissen wird es mit ensprechenden Lochungen in seinen Spalten (Kanälen) versehen. Die Lochung in Spalte 1 heißt Kanal 1, die in Spalte 2 Kanal 2 usw. Die Lochungen sind immer in der Höhe der gewünschten Zeile anzubringen, die auf einen größeren Zwischenraum folgt: Die Datumszeile beginnt bei Kanal 1, die Kundenanschrift bei Kanal 2, die erste Artikelzeile bei Kanal 3 und die End-Summe bei Kanal 12.
Nachdem das Lochband zu einer endlosen Schleife zusammengeklebt und an einer bestimmten Stelle des Druckers eingespannt wurde, läuft es synchron mit den einzelnen Formularblättern um.

Dabei hat man die Möglichkeit, durch entsprechende Programmbefehle den Formularvorschub zu steuern. Durch den Befehl

Vorschub auf Kanal 1

wird das Rechnungsformular sofort bis zur Datumszeile unter die Druckstation bewegt. Nach dem Ausdrucken des Tagesdatums (im Programmablaufplan nach Abb. 170 nicht berücksichtigt) bewirkt der Befehl

Vorschub auf Kanal 2

einen Formularvorschub bis zur Zeile für die Kundenanschrift.
Folgt im Programm nach einem Druckbefehl kein Kanalhinweis, so bleibt es bei dem üblichen 1-Zeilen-Vorschub.
Unter Berücksichtigung dieser Tatsache läßt sich jetzt das Bedrucken des Rechnungsformulares (Abb. 148) wie folgt aufgliedern:

Vorschub auf Kanal 1
Datumszeile drucken
Vorschub auf Kanal 2
Kundenanschrift (Teil A) drucken
Kundenanschrift (Teil B) drucken
Kundenanschrift (Teil C) drucken
Vorschub auf Kanal 3
Erste Artikelzeile drucken
Zweite Artikelzeile drucken
Dritte Artikelzeile drucken
. . . .
. . . .
. . . .
Vorschub auf Kanal 12
End-Summe drucken

(Nach Ausgabe der End-Summe müßte das Programm wieder zu dem Befehl

Vorschub auf Kanal 12
End-Summe drucken

für das nachfolgende Rechnungsformular verzweigen).
Auf diese Art und Weise erübrigt die Lochbandsteuerung das lästige Ausdrucken von Leerzeilen; das Programm kann in bedeutend kürzerer Zeit beendet werden.
(In manchen modernen Druckern ist das Lochband zur Steuerung des Formularvorschubes nicht mehr vorhanden. Es ist durch entsprechende Programmroutinen ersetzt.)
Der in diesem Kapitel beschriebene Kettendrucker stellt natürlich nicht die einzige Möglichkeit für eine Druckausgabe dar. Die Vielfalt anderer Druckprinzipien setzt einen in Erstaunen, so daß einige von ihnen hier zumindest nicht unerwähnt bleiben sollen:
Bei Nadeldruckern werden die Zeichen punktartig zusammengesetzt. Der Erzeugung der einzelnen Punkte dienen feine Drucknadeln, deren Spitzen gegen Farbband und Aufzeichnungsträger geschlagen werden.
Laser-Drucker haben phantastisch hohe Druckleistungen:

1,2 Millionen Zeilen pro Stunde

Bei ihnen wird die auszugebende Information von einem Laserstrahl auf eine sich drehende besonders beschichtete Trommel geschrieben, von welcher ein Abdruck auf den Aufzeichnungsträger erfolgt.
Für Thermodrucker ist die Verwendung wärmeempfindlichen Papiers erforderlich. Zur Zeichendarstellung wird an entsprechenden Stellen des Papiers Wärmeenergie aufgebracht, wodurch eine Verfärbung des Papiers bewirkt wird und die Zeichen sichtbar werden.
Tintenstrahldrucker gelangen auch immer mehr in den Blickpunkt des Interesses: An einer Düse wird eine Vielzahl allerkleinster Tintentröpfchen (ca. 100 000 Tröpfchen pro Sekunde) erzeugt, die anschließend in einem ablenkbaren Strahl auf das Papier geleitet werden. Dieser Strahl ist so fein, daß er die feinsten Zeichenkonturen erzeugen kann.
Stahlbanddrucker unterliegen einem ähnlichen Prinzip wie die Kettendrucker. Bei ihnen befinden sich die Zeichentypen auf einem umlaufenden Stahlband, von dem aus sie durch Druckhämmer zum Abdruck gelangen.

20. Magnetband-Einheiten

Das Wort Magnetband ruft sicherlich die Erinnerung an ein Tonbandgerät wach; man denkt unwillkürlich an seine Lieblingsschlager oder gar die „eigenen" Worte. Und die Frage nach der Aufzeichnung und Wiedergabe der Töne beantwortet sich fast von selbst; man spricht in ein Mikrophon, welches elektrische Spannungen erzeugt. Diese Spannungen werden einem Sprechkopf zugeführt, unter dem sich das Magnetband fortbewegt. Dabei erfolgt eine dem Ton entsprechende Magnetisierung des Bandes.
Liegt der Wunsch nahe, die eine oder andere Stelle des Bandes abzuhören, so ist dies beliebig oft möglich. Beim Abhören bewegt sich das Band unter einem Wiedergabekopf hindurch, von dem je nach der Magnetisierung der aufgezeichneten Töne elektrische Spannungen erzeugt und einem Lautsprecher zugeführt werden.
So kann man sich anhören, was man Tags- oder Monate zuvor auf das Band aufnahm.
Außerdem ist es auch möglich, alte Aufzeichnungen zu löschen oder durch neue zu überspielen. Das gleiche Tonband ist also beliebig oft für Neuaufnahmen verwendbar.
Die Magnetband-Einheit in der Computertechnik hat vieles mit dem Tonbandgerät gemeinsam. Es gilt allerdings die Einschränkung, daß im Rahmen der elektronischen

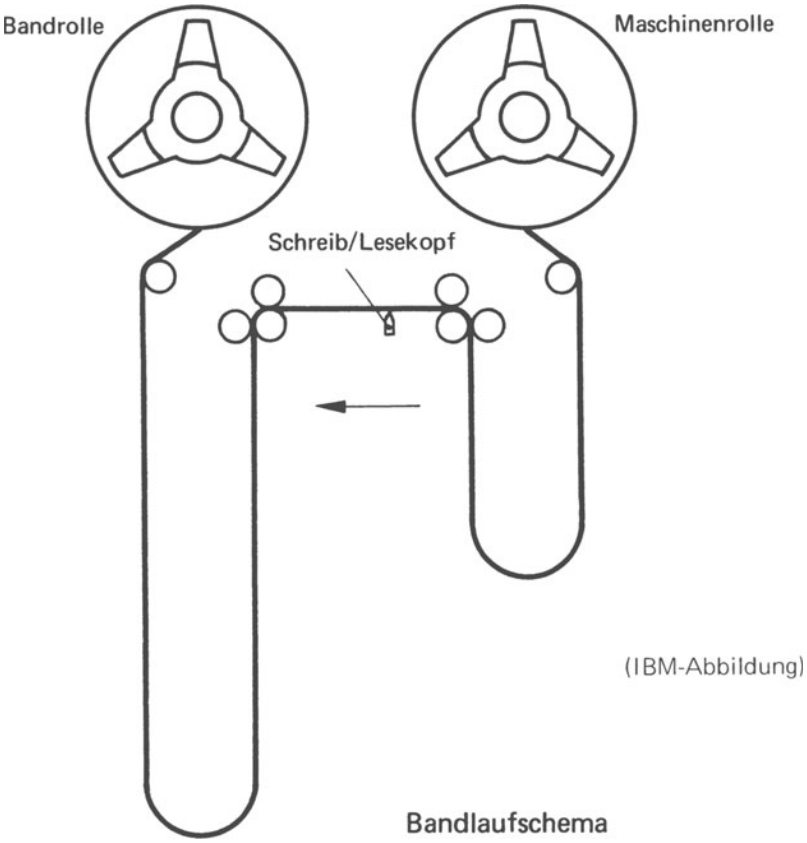
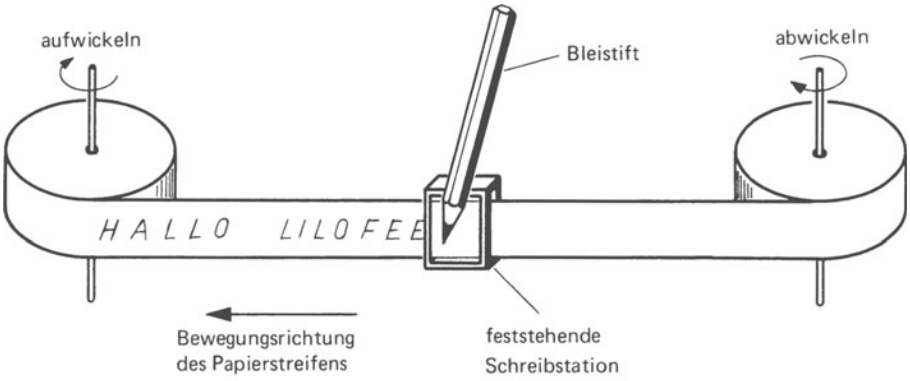


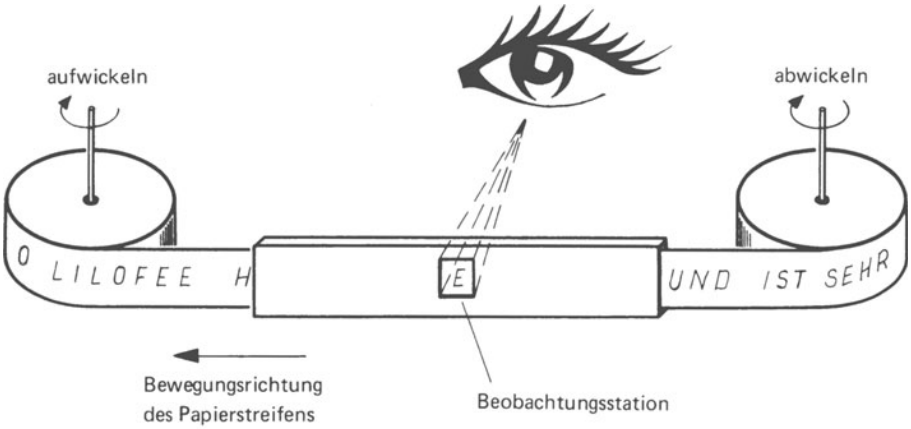
Abb. 149

Datenverarbeitung nicht etwa musikalische Darbietungen, sondern simple Daten aufzuzeichnen (zu schreiben) oder wiederzugeben (zu lesen) sind.
In einem einfachen Vergleich könnte man sich das Schreiben oder Lesen von Banddaten wie in Abb. 150 und Abb. 151 gezeigt vorstellen.



Schreibvorgang

Abb. 150



Lesevorgang

Abb. 151

Bei einer Magnetband-Einheit sind jedoch der Papierstreifen durch das Magnetband und der Bleistift bzw. das Auge durch den Schreib-Lese-Kopf (S-L-Kopf) ersetzt (Abb. 152). Wie in Abb. 137 gezeigt, treten die Daten auf dem Magnetband in Form verschlüsselter Zeichen (Ziffern und Buchstaben) auf. Jedem Zeichen ist eine ganz bestimmte Kombination von „Magnetpunkten" zugeordnet, die der S-L-Kopf ausgeben oder erkennen (Lesen) kann. Doch woher erhält nun der Schreibkopf die aufzuzeichnenden Daten mitgeteilt?
Er erhält sie von der Zentral-Einheit des Computers, mit dem die Magnetband-Einheit verbunden ist.
Die Zentral-Einheit (CPU = Central Processing Unit) enthält den Kernspeicher zur Programmspeicherung und einen elektronischen Mechanismus zur Ausführung der einzelnen Instruktionen.
Es ist Aufgabe des Programmes, die auf dem Magnetband aufzuzeichnenden Daten bereitzustellen und zwar in einem eigens dafür vorgesehenen Bandausgabebereich des Hauptspeichers (Abb. 153).
(Das stellt prinzipiell nichts Neues dar, denn selbst einem Drucker oder Karten-Stanzer wurden schon entsprechende Ausgabebereiche zugeordnet).
Sobald der Inhalt des Bandausgabebereiches als „Datensatz" auf das Magnetband ausgegeben werden soll, ist der Schreibbefehl

```
PUT     BAND
        └──── gewählter Dateiname für die Datensätze auf dem
              Magnetband
```

erforderlich: Das Band setzt sich dann ruckartig in Bewegung und der Schreibkopf tritt in Aktion. Er erzeugt nacheinander für jedes Zeichen des Bandausgabebereiches die zugehörige Magnetpunktkombination.
Abb. 137 zeigt unter anderem die Magnetpunktkombinationen (quer zur Bewegungsrichtung des Bandes) für die Zeichen „O" (bitweise Darstellung 1111 1111), „7", „A" und „? ".
Eine binäre „1" ist durch einen vorhandenen, eine binäre „0" durch einen fehlenden Magnetpunkt gekennzeichnet. Die Aufzeichnung eines Zeichens erfolgt in der unvorstellbar kurzen Zeit von ca. 0,0003 Sekunden.
Im Anschluß an die Aufzeichnung des ersten Zeichens folgt die des zweiten, des dritten usw., bis der gesamte Inhalt des Bandausgabebereiches als Datensatz auf dem Magnetband steht. Dann wird das Band in seiner Bewegung ruckartig abgestoppt.
Bekäme ein Programmierer die Aufgabe gestellt, Lochkartendaten auf einem Magnetband zu speichern, so wären dazu im Prinzip nur drei Instruktionen erforderlich.

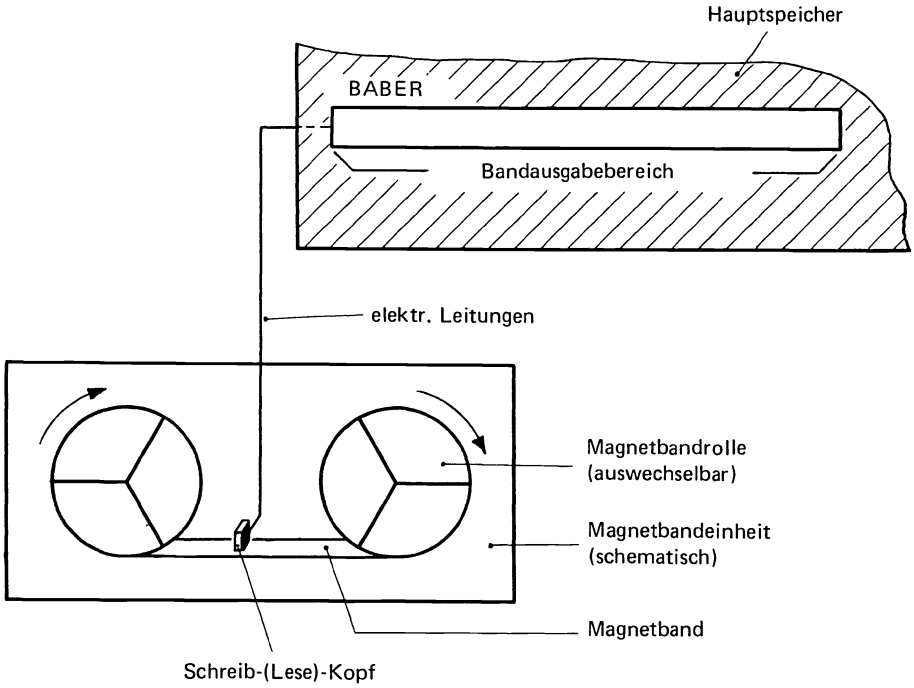


Abb. 152

```
ANF     GET     KART
        MVC     BABER,KARBER
        PUT     BAND
        (B      ANF)
```

(Für die Lochkarten wurde der Dateiname KART, für die Bandsätze der Dateiname BAND vereinbart. Der Karteneinlesebereich ist mit KARBER, der Bandausgabebereich mit BABER bezeichnet.
Nach dem Einlesen der ersten Lochkarte wird der Inhalt des Karteneinlesebereiches in den Bandausgabebereich übertragen. Anschließend erfolgt durch den Befehl

```
        PUT     BAND
```

die Ausgabe des gesamten Inhaltes des Bandausgabebereiches auf das Magnetband. Der dort erscheinende erste Datensatz entspricht also den Daten der zuvor eingelesenen Lochkarte. Sollen weitere Datensätze für andere Lochkarten angelegt werden, so muß das Programm immer wieder zu der Instruktion

```
ANF     GET     KART
```

zurückkehren.

ANF	GET	KART
	MVC	BABER, KARBER
	PUT	BAND
	(B	ANF)

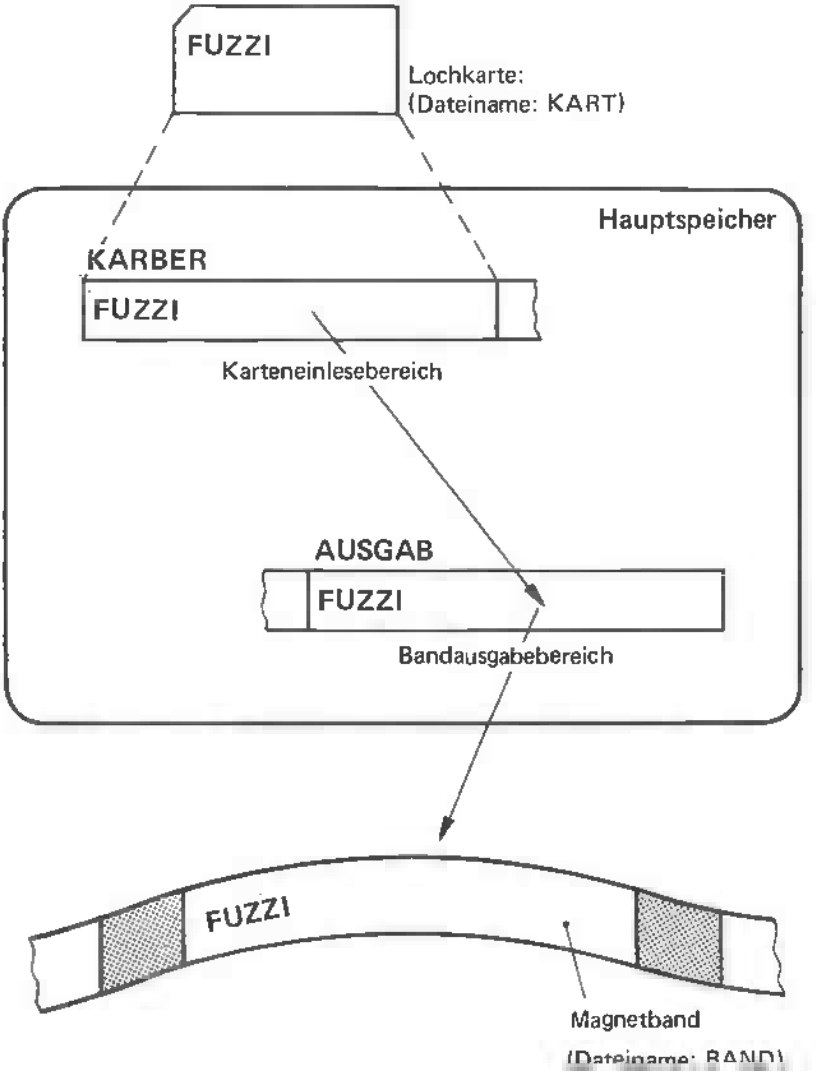


Abb. 153

Im Verlaufe des Programmes wird Satz für Satz streng aufeinanderfolgend (sequentiell) auf das Magnetband ausgegeben. Man spricht in diesem Zusammenhang von einem

sequentiell (fortlaufend) gespeicherten Datenbestand.

Dieses Beispiel soll jedoch nicht den Eindruck erwecken, daß sich das Anlegen einer Band-Datei in der Übernahme von Lochkartendaten erschöpft. Die Informationen für einen Bandsatz können von den verschiedenen Aufzeichnungsträgern stammen. Darüberhinaus ist es oft erforderlich, die eingelesenen Daten durch entsprechende Programmbefehle zu verändern und zu verarbeiten, bevor sie in den Bandsatz aufgenommen werden können.
Der kombinierte Schreib-Lesekopf (S-L-Kopf, Abb. 149) ist so konstruiert, daß er nicht nur für Schreib- sondern auch für Leseoperationen verwendet werden kann. Nach Erteilung eines Lesebefehles (per Programm) bewegt sich das Band in der Länge eines Datensatzes an dem (S)-L-Kopf vorbei. Dabei wird der Inhalt des Bandsatzes gelesen und in die entsprechenden Stellen des Bandeinlesebereichs des Hauptspeichers übertragen. Anschließend bewegt sich das Band nicht weiter; es setzt sich erst dann wieder in Bewegung, wenn erneut eine Lesebefehl erteilt wird (der „alte" Inhalt des Bandeinlesebereiches geht dabei durch Überschreiben verloren).
Sollen nun im Rahmen eines Programmes die Sätze einer Banddatei gelesen und verarbeitet werden, so ist dies nur in einer ganz bestimmten Reihenfolge möglich. Ganz streng und ohne jede Ausnahme nach dem ersten Lesebefehl der erste, nach dem zweiten Lesebefehl der zweite, dann der dritte, vierte, fünfte Bandsatz usw. in den Hauptspeicher bis entweder das Bandende erreicht oder ein Datei-Ende-Kennzeichen automatisch festgestellt wird. Fachleute sprechen in diesem Zusammenhang von einer sequentiellen Bandverarbeitung. Danach ist es ausgeschlossen, daß man wahlweise – je nach Bedarf – zum Beispiel erst den 23., dann den 17. und darauf den 183. Bandsatz einlesen könnte. Das Magnetband kann nun einmal nicht (wie ein Tonband) zu irgendeiner Stelle vor- oder zurückgespult werden.
Bei der näheren Betrachtung eines einzelnen Bandsatzes läßt sich feststellen, daß für die Speicherung der 80 Zeichen ca. 1/4 cm Bandlänge erforderlich sind (das entspricht einer Zeichendichte von 320 Bytes/cm). Bei einer Kluftgröße (eine Kluft trennt einen Bandsatz von dem anderen) von 1/4 cm wäre demnach nur die Hälfte des Bandes mit Informationen beschrieben (Abb. 154). Diese Tatsache wird zurecht als großer Nachteil empfunden; denn das Band ist dadurch in seiner Speicherkapazität (Fassungsvermögen für Information) nicht gut ausgenutzt.
Einen Ausweg aus dieser Misere bietet die Methode der Blockung: Danach wird einfach der Inhalt mehrerer Lochkarten (zum Beispiel 3 Stück) zusammengefaßt und als Dreier-Block auf das Band ausgegeben (Abb. 155). Der Bandausgabebereich – wegen der Blockung Blockausgabebereich genannt – muß jetzt allerdings größer gewählt werden (3x80 = 240 Stellen), um jeweils drei Lochkartensätze aufnehmen zu können. Ein Blick auf Abb. 155 zeigt, daß jetzt das Magnetband kapazitätsmäßig viel besser ausgenutzt ist, weil sich durch die gewählte Dreierblockung (Blockungsfaktor 3) die Anzahl der Klüfte beträchtlich verringert hat.
Bei modernen Bandeinheiten gibt es Zeichendichten bis ca. 2500 Zeichen/cm und Lesegeschwindigkeiten bis ca. 1 000 000 Zeichen/sec.

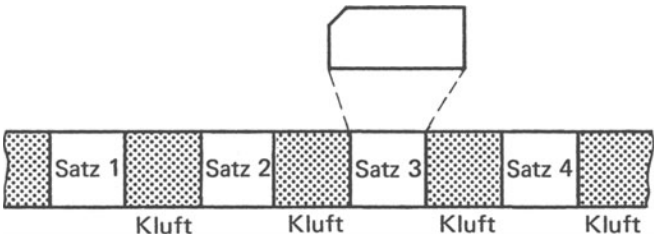


Abb. 154

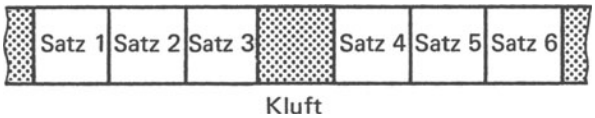


Abb. 155

In der Programmierung wird ein Block häufig auch als physischer Datensatz bezeichnet, der sich aus einzelnen logischen Sätzen zusammensetzt (Abb. 156).
Es ist verständlich, daß bei der Dreier-Blockung*) jeweils der Inhalt von drei nacheinander eingelasseen Lochkarten an die entsprechenden Stellen des Blockausgabebereiches des Hauptspeichers zu übertragen ist, bevor der Schreibbefehl für diesen Block erteilt werden darf. Diese Forderung würde natürlich zusätzlichen Programmieraufwand bedingen, gäbe es nicht die Ein/Ausgabe-Steuerung — sie ist durchaus kein besonderer Mechanismus in dem Computer. Sie verkörpert mehr jene Eigenschaft, bestimmte pauschale Angaben in der symbolischen Dateibeschreibung in eine Folge einfacher Maschinenbefehle aufzugliedern. Daß diese Angaben irgendwie im Zusammenhang mit der Ein-/Ausgabe von Daten stehen, verrät schon der Name Ein/Ausgabe-Steuerung.

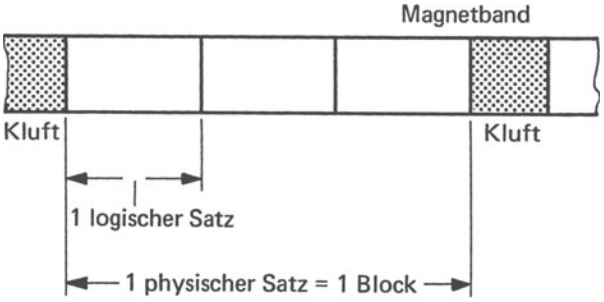


Abb. 156

* Es sind auch andere Blockungsfaktoren bis zu 100 und mehr möglich.

So könnte zum Beispiel das EOF-Merkmal ENDBAN (Name der Instruktion, zu der das Programm verzweigen soll, wenn kein Bandsatz mehr eingelesen werden kann, wenn also das Ende der Band-Datei erreicht ist) in folgende echte Instruktionen aufgelöst werden: (Aus Gründen des Verständnisses sind diese Instruktionen hier in wörtlicher Auslegung und nicht in codierter Darstellung wiedergegeben):

(Bandsatz lesen)
Ist kein Bandsatz mehr auf dem Band enthalten?
Verzweige nach ENDBAN bei Antwort ja

Und wie glücklich kann sich der Programmierer erst schätzen, daß die Ein/Ausgabe-Steuerung auch die Blockung von Datensätzen unterstützt. Er braucht in der Dateibeschreibung für die betreffende Magnetband-Einheit nur folgende Angaben zu machen:

Name der Datei	BAND
Blocklänge	3 x 80
Blockungsfaktor	3
Name des Block-ausgabebereiches	BABLO
Name des Arbeits-bereiches	ARBEI
EOF-Merkmal	ENDBAN
Art der Datei	OUTPUT

Dadurch bleibt ihm die Zusammenstellung der Lochkartendaten zu einem Block durch entsprechende Programmbefehle erspart.
In der Dateibeschreibung wurde BABLO als symbolischer Name für den Blockausgabebereich gewählt. Die Größe dieses Bereiches muß durch die Definitions-Instruktion

```
BABLO    DS        CL240
```

festgelegt werden.
ARBEI ist der symbolische Name eines Hauptspeicherbereiches, der als sogenannter „Arbeitsbereich" Verwendung finden soll. Der Arbeitsbereich hat die Länge eines logischen Satzes, umfaßt in diesem Falle also 80 Bytes und ist dementsprechend zu definieren:

```
ARBEI    DS        CL80
```

Mit Hilfe dieser Angaben vereinfacht sich das Erstellen dreifach-geblockter Bandsätze aus Lochkarten wie folgt: Durch den Befehl

```
PUT      BAND,ARBEI
```

symbolischer Name des Arbeitsbereiches
symbolischer Dateiname

wird der Inhalt des Arbeitsbereiches (dieser soll zunächst die Daten der zuerst eingelesenen Lochkarte enthalten) in die erste Sektion des Blockausgabebereiches übertragen

(Abb. 157). Nach dem Einlesen der zweiten Lochkarte und der Übertragung des Inhaltes des Karteneinlesebereiches in den Arbeitsbereiche kommt der Befehl

```
PUT        BAND,ARBEI
```

erneut zur Ausführung. Er bewirkt diesmal die automatische Übertragung des neuen Inhaltes des Arbeitsbereiches (zweite Lochkarte) in die zweite Sektion des Blockausgabebereiches.

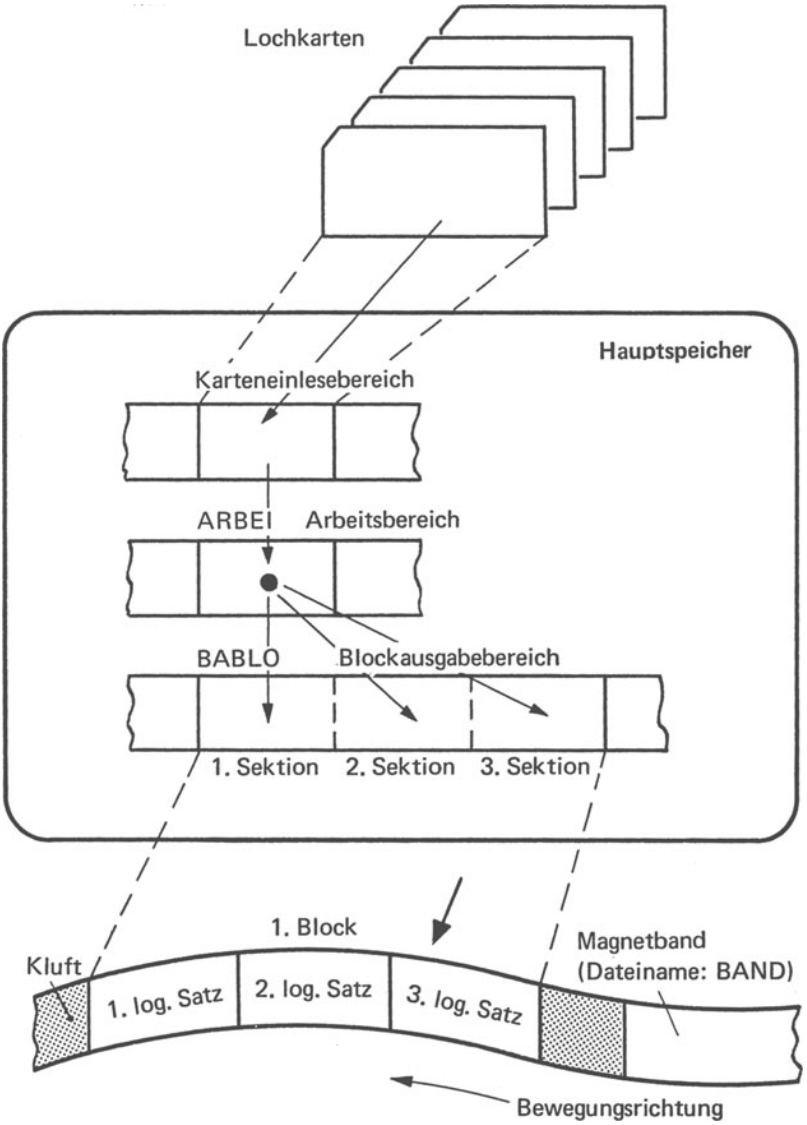


Abb. 157

Entsprechendes geschieht auch nach dem Einlesen der dritten Lochkarte. Sobald der Datenblock im Blockausgabebereich BABLO komplett zusammengestellt ist – also nach dreimaliger Ausführung des Befehles

```
PUT      BAND,ARBEI
```

wird dieser Block automatisch ohne Anwendung einer besonderen Programminstruktion auf das Band geschrieben.
Und nach dem Einlesen der vierten Lochkarte etc. beginnt erneut die Zusammenstellung eines zweiten Blockes.
Die Ein/Ausgabe-Steuerung übernimmt nicht nur die Zuweisung eines logischen Satzes in die richtige Sektion des Blockausgabebereiches, sie bestimmt auch den Zeitpunkt des Ausschreibens des Blockes auf das Band. Der Programmierer braucht sich deshalb nur auf den Befehl

```
PUT      BAND,ARBEI
```

zu beschränken.
Das Programm zur Erstellung dreifach geblockter Bandsätze aus Lochkarten würde also im wesentlichen lediglich aus den Befehlen

```
ANF      GET      KART
         MVC      ARBEI,KARBER
         PUT      BAND,ARBEI
         (B       ANF)
```

bestehen.
Auch beim Einlesen geblockter Bandsätze kann man sich eines Arbeitsbereiches bedienen (Abb. 158).
Es ist natürlich der Wunsch eines jeden Programmierers, dabei mit möglichst wenig Aufwand auszukommen: Durch den Lesebefehl

```
GET      BAND,ARB
```

symbolischer Name des Arbeitsbereiches
Dateiname

wird jeweils ein einzelner logischer Satz im Arbeitsbereich ARB zur Verfügung gestellt. Alles andere erfolgt automatisch unter Steuerung des IOCS:
Das Einlesen der Blöcke in den Blockeinlesebereich des Hauptspeichers und die schrittweise Übertragung der dort stehenden logischen Sätze in den Arbeitsbereich.
Bei der gewählten Dreier-Blockung ergeben sich also bei mehrfacher Ausführung des Befehles

```
GET      BAND,ARB
```

die in Abb. 159 aufgeführten Schritte.

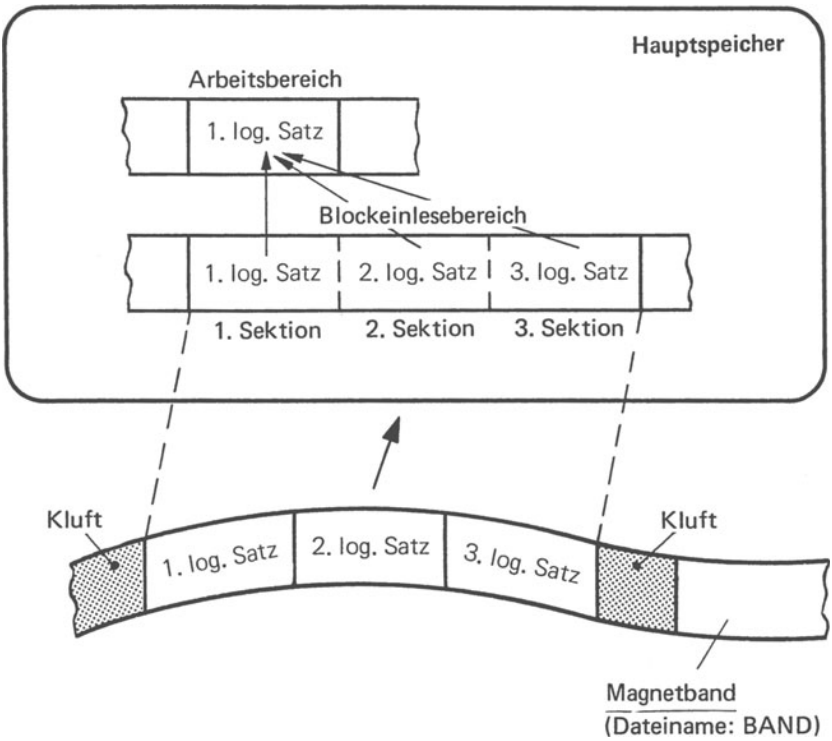
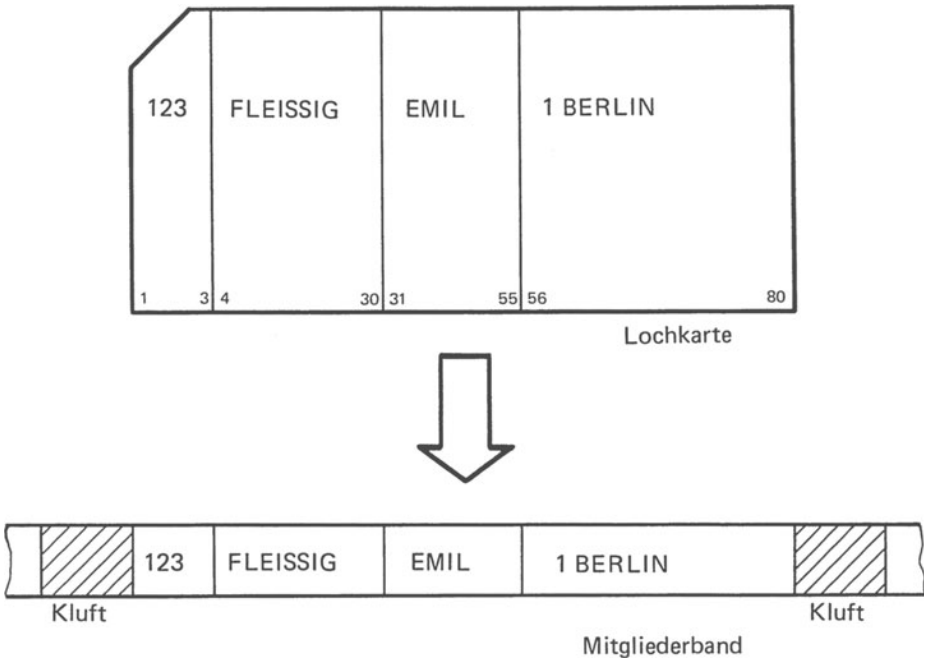


Abb. 158

Wie oft wurde der Lesebefehl GET BAND, ARB erteilt?	Welcher log. Satz steht nach Ausführung dieses Befehles im Arbeitsbereich?	Welcher Datenblock steht nach Ausführung dieses Befehles im Blockeinlesebereich?	Aus welcher der drei Sektionen des Blockeinlesebereiches stammt der in den Arbeitsbereich übertragene logische Satz
1 mal	1 log. Satz	1. Block	1. Sektion
2 mal	2. log. Satz	1. Block	2. Sektion
3 mal	3. log. Satz	1. Block	3. Sektion
4 mal	4. log. Satz	2. Block	1. Sektion
5 mal	5. log. Satz	2. Block	2. Sektion
6 mal	6. log. Satz	2. Block	3. Sektion
7 mal	7. log. Satz	3. Block	1. Sektion
8 mal	8. log. Satz	3. Block	2. Sektion
.	.	.	.
.	.	.	.
.	.	.	.

Abb. 159

Die vorstehenden Bemerkungen zur sequentiellen Speicherungsform und zur sequentiellen Verarbeitungsform von Band-Dateien lassen sich am besten an Hand eines weiteren Beispieles erhärten. Dabei wird davon ausgegangen, daß eine Magnetband-Datei mit sogenannten Mitgliedersätzen existiert (Abb. 160). Diese Daten sollen von Lochkarten geliefert worden sein.



Feldeinteilung

Lochkarte:	Spalte	1 – 3	Mitgliedsnummer
	Spalte	4 – 30	Nachname
	Spalte	31 – 55	Vorname
	Spalte	56 – 80	Beruf
Bandsatz:	Stelle	1 – 3	Mitgliedsnummer
	Stelle	4 – 30	Nachname
	Stelle	31 – 55	Vorname
	Stelle	56 – 80	Beruf

Übernahme von Lochkartendaten auf Magnetband

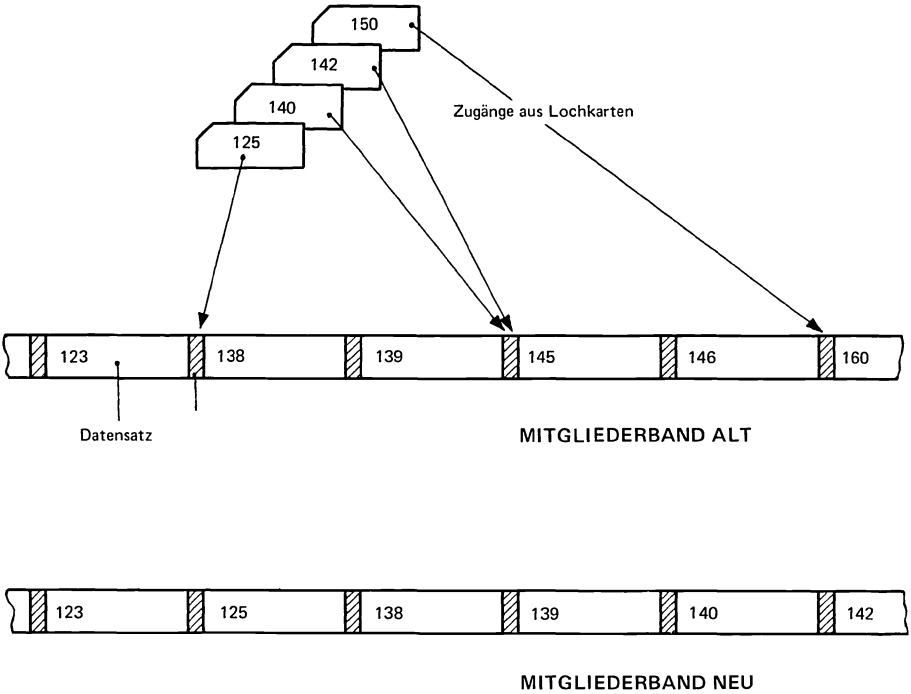
Abb. 160

Es ist einleuchtend, daß eine solche Band-Datei „gepflegt", d. h. auf dem neuesten Stand gehalten werden muß. Schließlich ergeben sich im Verlaufe der Zeit für einzelne Mitglieder Änderungen, sei es in der Adresse infolge eines Umzuges oder im Familienstand infolge einer Heirat etc. Mitunter kann auch eine Mitgliedschaft erlöschen (bei

Austritt, Ausschluß oder Todesfall) — in einem solchen Fall müßte der entsprechende Datensatz sogar „gestrichen" werden, während er bei einer bloßen Änderung nur auf den neuesten Stand zu bringen wäre.
Letztlich ist es auch denkbar, daß neue Mitglieder gewonnen werden, für die dann (neue) Datensätze anzulegen sind. Eine derartige Bestandspflege erfordert entsprechende Programme:

1. ZUGÄNGE
2. ABGÄNGE
3. ÄNDERUNGEN.

Zur Erläuterung der typischen Besonderheiten einer Bandverarbeitung dient hier das Programm „ZUGÄNGE":
Es wird angenommen, daß die Datensätze der bereits bestehenden Mitglieder-Datei („Mitgliederband alt") in aufsteigender Reihenfolge nach den Mitgliedsnummern auf dem Magnetband gespeichert sind (Abb. 161).
Aufsteigende Reihenfolge bedeutet allerdings nicht, daß sich die Mitgliedsnummern benachbarter Sätze nur um 1 unterscheiden dürfen, vielmehr sind größere Differenzen denkbar, jedoch muß die folgende Mitgliedsnummer stets größer als die vorausgehende sein.

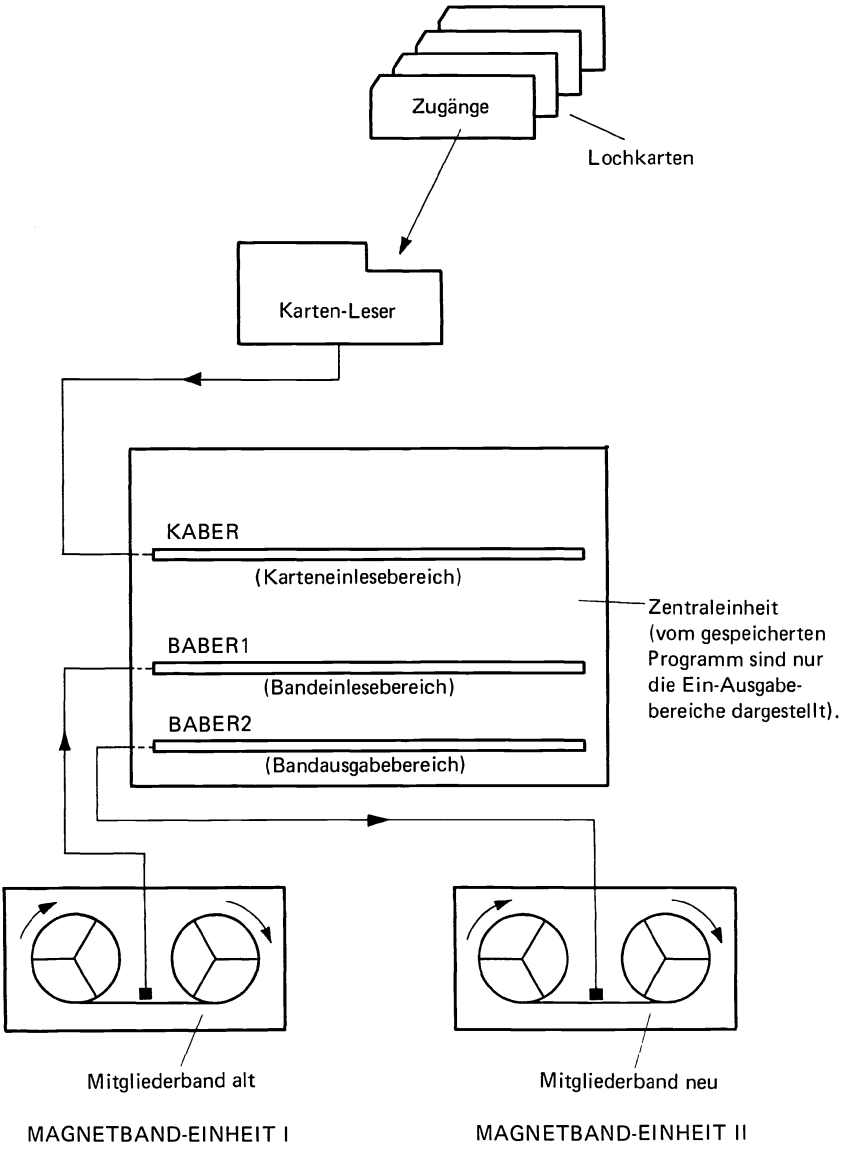


(zur Vereinfachung der Darstellung sind die Datensätze nur durch Angabe der Mitgliedsnummer gekennzeichnet).

Beispiele einer Band-Fortschreibung

Abb. 161

Als Zugänge sind neue Datensätze zwischen bereits bestehende einzufügen (Abb. 161), da die aufsteigende Rangordnung aus organisatorischen und programmtechnischen Gründen unbedingt erhalten bleiben muß. Damit ist jedoch die Aufgabe des Programmes „ZUGÄNGE" bereits klar umrissen; die Lösung erscheint einfacher als vielleicht erwartet:
Da man ein Band nicht einfach auseinanderschneiden wird, um an entsprechender Stelle einen neuen Satz einzufügen, bedient man sich eines zweiten Magnetbandes in einer



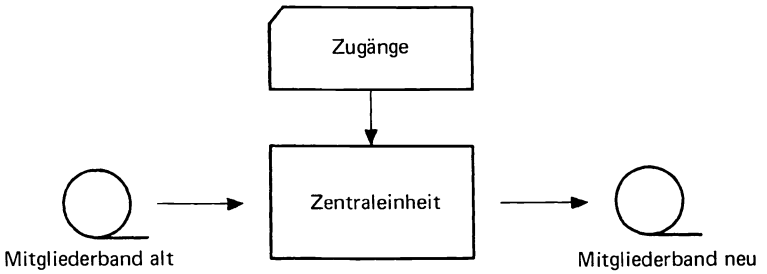
Schema einer Band-Fortschreibung

Abb. 162

anderen Band-Einheit. Auf dieses Band werden dann alle Datensätze unter gleichzeitiger Berücksichtigung der Zugänge geschrieben (Abb. 162). Einen solchen Vorgang nennt man Band-Fortschreibung.
Unter der Annahme, daß die Daten für die Zugänge aus Lochkarten stammen, sind somit drei Dateien zu berücksichtigen:

a) die Lochkarten auf der Kartenabfühleinheit
b) die Bandsätze auf der Magnetband-Einheit I
c) die Bandsätze auf der Magnetband-Einheit II

Die Datensätze des „Mitgliederbandes alt" und der Lochkarten werden, so wie es das Programm vorsieht, in den Hauptspeicher eingelesen und in der gewünschten Reihenfolge auf das „Mitgliederband neu" ausgegeben (Abb. 162).
Die graphische Darstellung in Abb. 162 läßt sich zu Abb. 163 vereinfachen, wenn für Lochkarten und Magnetband die in der elektronischen Datenverarbeitung üblichen Symbole verwendet werden.



Mitgliederbestand fortschreiben

Abb. 163

Die Lochkarten sind nach aufsteigender Mitgliedsnummer sortiert der Verarbeitung zuzuführen.
Beim Erstellen eines Programmablaufplanes für die Bandfortschreibung des Mitgliederbestandes ist folgendes zu beachten:
Nach dem Einlesen einer Lochkarte steht der Datensatz für ein neugewonnenes Mitglied im Karteneinlesebereich KABER.
Anschließend muß vom „Mitgliederband alt" ein Bandsatz in den Bandeinlesebereich BABER1 eingelesen werden. Dann ist zu entscheiden, welcher der beiden Datensätze (in KABER bzw. in BABER1) die niedrigere Mitgliedsnummer hat. Sobald dies ermittelt ist, wird der Datensatz mit der niedrigeren Mitgliedsnummer in den Bandausgabebereich BABER2 übertragen und durch einen Schreibbefehl auf das „Mitgliederband neu" ausgegeben. Danach ist entweder ein neuer Bandsatz vom „Mitgliederband alt" in den Bandeinlesebereich BABER1 oder eine neue Lochkarte in den Karteneinlesebereich KABER einzulesen und durch einen Vergleich festzustellen, welcher Satz nunmehr mit der nächsthöheren Mitgliedsnummer auf das „Mitgliederband neu" auszugeben ist.
Der Programmablauf hat zu gewährleisten, daß alle Datensätze, unabhängig von ihrer Herkunft („Mitgliederband alt" oder Zugänge aus Lochkarten) immer nach aufsteigender Reihenfolge auf das „Mitgliederband neu" geschrieben werden (Abb. 161).

Typisch für eine solche Bandfortschreibung ist also, daß die Sätze eines Bandes Satz für Satz gelesen werden (sequentielle Verarbeitung) und unter Berücksichtigung der Zugänge eine neue Banddatei durch aufeinanderfolgende Ausgabe neuer Bandsätze erstellt wird (in sequentieller Speicherungsform).

21. Magnetplatten-Einheiten

Seit ihrer Einführung hat die Magnetplatte in der Computertechnik eine ständig zunehmende Bedeutung erfahren. Sie ist das markanteste Beispiel eines Datenspeichers mit direktem Zugriff (direct access). Der Schreib-Lese-Kopf (S-L-Kopf) kann über der ständig rotierenden Platte sprungartig, in Bruchteilen von Sekunden, seine Position ändern. Er ist jeweils auf eine der konzentrisch angeordneten Datenspuren ausgerichtet (Abb. 164). Für seine Einstellung ist das Vorliegen der Nummer der gewünschten Datenspur erforderlich. Eine solche Angabe wird, ausgelöst durch einen entsprechenden Programmbefehl, von der Zentral-Einheit an einen Zugriffsmechanismus signalisiert. Dieser bewirkt dann die Einstellung des S-L-Kopfes.

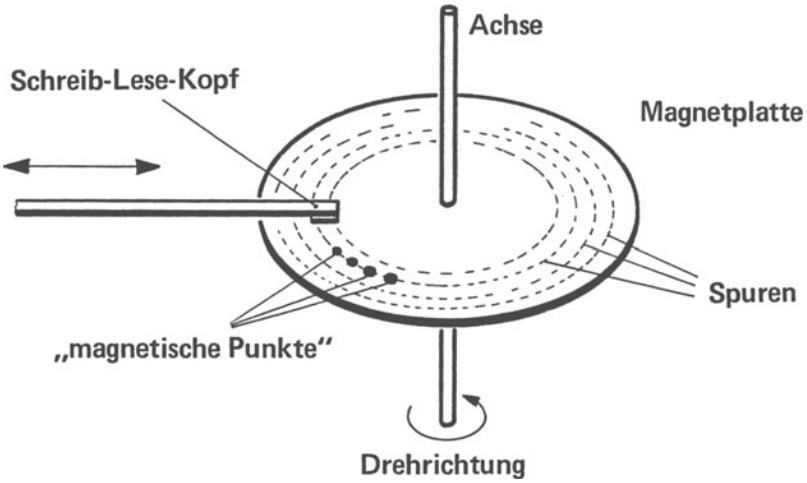
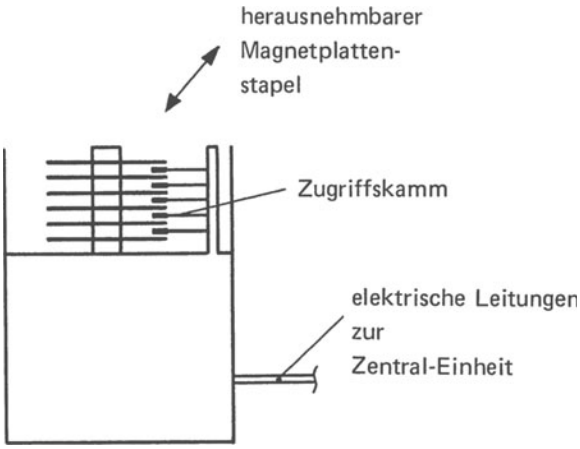


Abb. 164

Bei einem Schreibbefehl werden die Zeichen des Plattenausgabebereiches nacheinander in Form kleiner magnetischer Punkte auf die Platte ausgegeben. (Nur in den seltensten Fällen ist ein Datensatz so lang, daß er eine ganze Spur ausfüllt. Zumeist haben deshalb innerhalb einer Spur mehrere Sätze Platz).
Ein Lesebefehl hingegen veranlaßt den S-L-Kopf zum Einlesen eines Datensatzes von der Magnetplatte in den Platteneinlesebereich des Hauptspeichers. Um welchen Datensatz es sich dabei im einzelnen handeln soll, ist jeweils im Programm mit anzugeben. Es wäre durchaus denkbar, erst den 2. Satz der äußersten Spur, anschließend den 5. Satz der zweitinnersten Spur und dann den 1. Satz der viertäußersten Spur einzulesen; ein völlig willkürliches Vorgehen, wie es beim Magnetband nicht möglich war: Ein typisches Beispiel für einen direkten Zugriff (direct access). Beim Magnetband konnten die einzelnen Sätze nur in streng aufeinanderfolgender Reihenfolge – Satz Nr. 1, Satz Nr. 2, Satz Nr. 3, Satz Nr. 4 etc. – also sequentiell verarbeitet werden.
Von einer solchen Einschränkung ist man bei der Magnetplatte aufgrund des beweglichen S/L-Kopfes befreit. Dieser kann sich jederzeit auf einen x-beliebigen Plattensatz einstellen, oder, um es genauer zu sagen, auf jede x-beliebige Spur. Die Sätze einer Spur

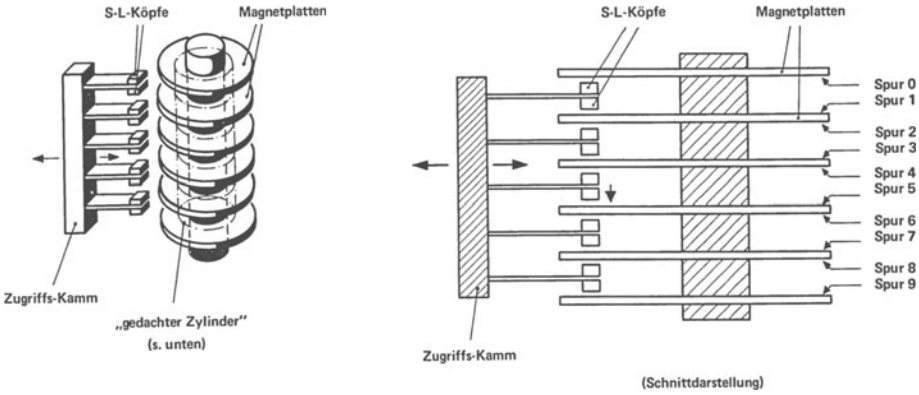
bewegen sich nacheinander am S/L-Kopf vorbei. Eingelesen wird jedoch immer nur der gewünschte Satz, im richtigen Moment.

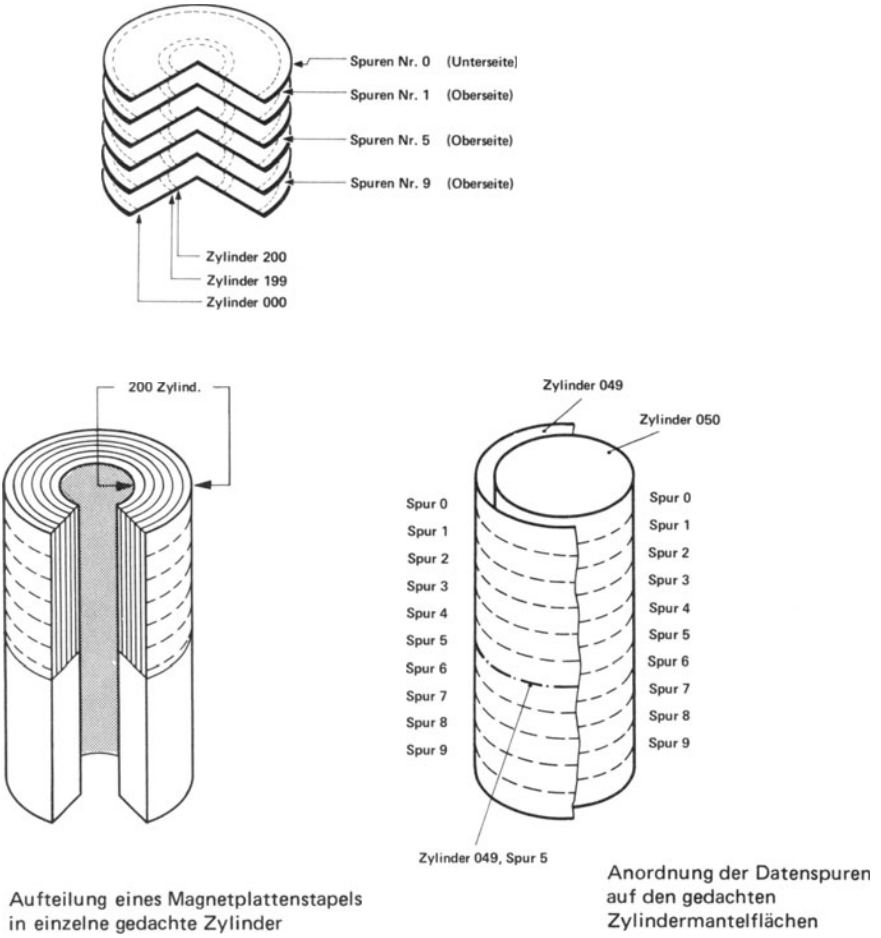
Zur Speicherung größerer Datenmengen reicht eine einzige Magnetplatte nicht aus. Aus diesem Grunde hat sich die Verwendung von Magnetplattenstapeln (Abb. 165) durchgesetzt. Sie bestehen aus übereinanderliegenden Magnetplatten, die auf einer gemeinsamen Achse angebracht sind. Jeder Magnetplattenstapel wird bei Bedarf in eine Magnetplattenspeicher-Einheit eingesetzt, dort steht er während eines Programmlaufes zur Datenein- bzw. Datenausgabe zur Verfügung.



Magnetplattenspeicher-Einheit

Abb. 165





Schematische Darstellung eines Magnetplattenstapels

Abb. 166

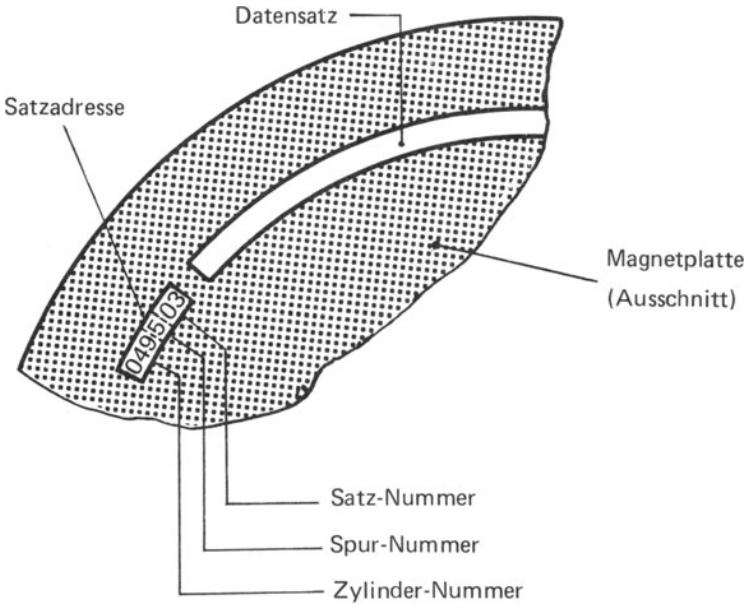
Mit Ausnahme der beiden äußeren, durch Kratzer gefährdeten Grund- und Deckflächen, tragen alle Plattenoberflächen eine Bezeichnung: Spur 0, Spur 1, Spur 2, . . ., Spur 9 (Abb. 166)*). Jeder dieser Plattenoberflächen ist ein S-L-Kopf zugeordnet. Die Gesamtheit aller S/L-Köpfe bildet einen gemeinsamen starren „Zugriffskamm", der sich in Richtung der Plattenachse verschieben kann. Durch diese Konstruktion bedingt sind sämtliche S/L-Köpfe immer auf alle untereinander liegenden Datenspuren ausgerichtet, auf die Spuren eines gedachten Zylinders (Abb. 166). Jeder dieser Zylinder trägt eine Nummer (z. B. von 000 bis 200).

*) Um eventuellen Irrtümern vorzubeugen, sei darauf hingewiesen, daß zum Beispiel die Bezeichnung Spur 5 allen Datenspuren der 5. Plattenoberfläche gemeinsam ist.

Gilt es, einen ganz bestimmten Datensatz innerhalb des Plattenstapels einzulesen, so sind dazu drei Angaben – die sogenannte „Hausadresse" – erforderlich:

Die Zylinder-Nummer
Die Spur-Nummer
Die Satz-Nummer

Die Zylinder-Nummer bewirkt die Einstellung des Zugriffskammes auf den genannten Zylinder; anschließend nähert sich der durch die Spur-Nummer bezeichnete S/L-Kopf der entsprechenden Plattenoberfläche, um im geeigneten Moment den gewünschten Datensatz einzulesen.
Die „Hausadresse" (Satzadresse) ist bei weitem keine fiktive Vorstellung; sie wird automatisch vor jeden einzelnen Datensatz auf die Magnetplatte geschrieben (Abb. 167).
Sie ist die Voraussetzung für das spätere Einlesen eines bestimmten Datensatzes in den Hauptspeicher. Vor dem Lesebefehl ist die genaue „Hausadresse" des gewünschten Satzes zu benennen und in ein ausgewiesenes Hauptspeicherfeld (Plattenadreßfeld) zu übertragen (Abb. 168).

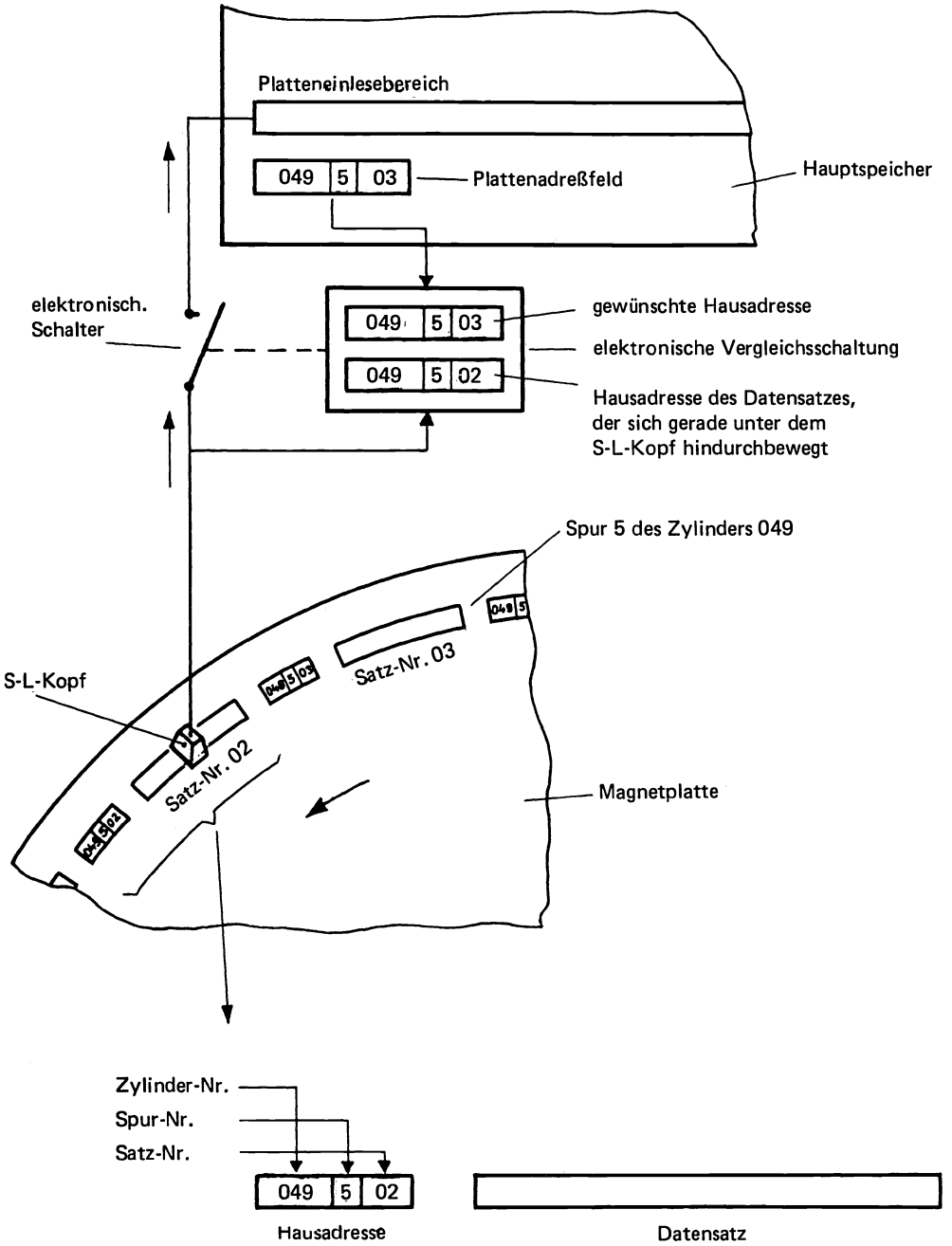


Kennzeichnung eines Datensatzes auf der Magnetplatte durch die Satzadresse

(Es handelt sich in diesem Falle um den Datensatz mit der Nr. 03 in der Spur 5 des Zylinders 049).

Abb. 167

Die Satz-Nummern in einer Spur zählen jeweils von 0 an aufwärts.



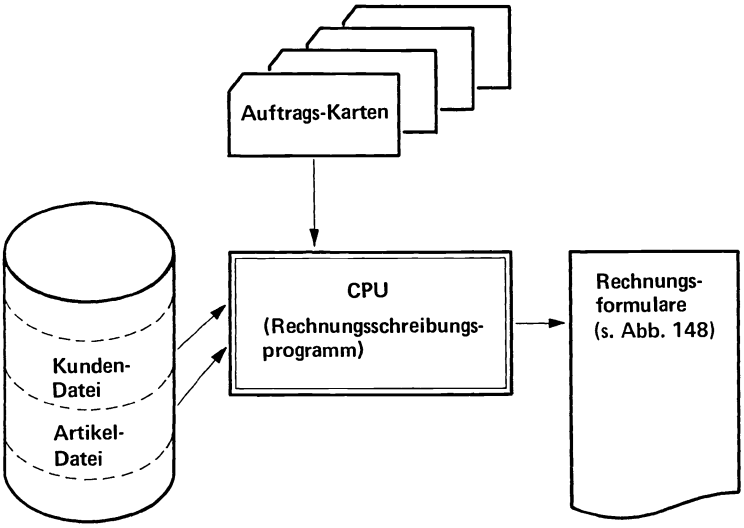
Vorgänge beim Einlesen eines Plattensatzes

Abb. 168

Mit Hilfe der Werte im Plattenadreßfeld erfolgt die Einstellung des Zugriffsmechanismuses: Der Schreib-Lese-Kopf-Kamm rückt auf den genannten (049.) Zylinder vor und der durch die Spurnummer bezeichnete (5.) S-L-Kopf nähert sich seiner Plattenoberfläche. Da die Magnetplatte ständig rotiert, werden sämtliche Datensätze der ausgewählten Spur am S-L-Kopf vorbeibewegt. Es ist jedoch erwünscht, daß von ihnen nur ein einziger Satz (der mit der Satznummer 03) in den Platteneinlesebereich des Hauptspeichers gelangt. Diese Forderung wird auf folgende Art und Weise erfüllt:
In den elektrischen Leitungen für die Datenübermittlung vom S-L-Kopf zum Hauptspeicher ist ein elektronischer Schalter eingebaut (Abb. 168). Er schließt sich automatisch nur für eine ganz kurze Zeitspanne, in der sich gerade der Datensatz Nr. 03 unter dem S-L-Kopf hindurchbewegt.
Dieser Zeitpunkt wird von einer speziellen elektronischen Vergleichsschaltung (Abb. 168) bestimmt. Der Vergleich erstreckt sich auf zwei Werte: die im Plattenadreßfeld bereitgestellte Hausadresse des gewünschten Satzes und die jeweilige Hausadresse des Satzes, der sich gerade unter dem S-L-Kopf hindurchbewegt*). Sobald sich eine Übereinstimmung ergibt, schließt sich der Schalter in den Übertragungsleitungen und der Inhalt des gewünschten Datensatzes Nr. 03 gelangt in den Platteneinlesebereich des Hauptspeichers.
Mit dieser Methode des direkten Zugriffs, des gezielten Einlesens eines bestimmten Datensatzes, lassen sich viele Aufgaben in der elektronischen Datenverarbeitung besonders vorteilhaft lösen.
Das Zylinder-Konzept mag zunächst als ein recht willkürliches Orientierungsprinzip für den Plattenstapel empfunden werden. Doch bei genauerer Betrachtung offenbart sich eine nicht zu übersehende Zweckmäßigkeit:
Angenommen, eine Platten-Datei umfaßt 1125 fortlaufend hintereinander gespeicherte Datensätze.
Die räumliche Anordnung dieser Sätze innerhalb des Stapels unterliegt folgendem Schema: Zunächst füllen die Datensätze eine ganze Spur aus, zum Beispiel die Spur 0 des Zylinders 010. Daran schließen sich dann die nächsthöheren Spuren (1, 2, . . ., 9) des gleichen Zylinders 010 an. Und falls dieser Platz immer noch nicht für die ganze Datei ausreichen sollte, folgen die Spuren 0, 1, 2, 3 . . . des Zylinders 011 usw.
Wenn innerhalb einer Spur 25 Stück 80-stelliger Sätze untergebracht werden können, wären für die gesamte Datei (1125 Sätze) insgesamt 45 Spuren, also 4 Zylinder und 5 Spuren erforderlich (ein Zylinder umfaßt 10 Spuren). Mit dieser Abschätzung ist aber noch nicht die Lage der Datei innerhalb des Stapels fixiert, denn das erfolgt erst durch die Festlegung der Bereichsgrenzen: Der Untergrenze (z. B. Zylinder 010, Spur 0) und

* Der S-L-Kopf liest alle Daten einer Spur, so auch die jedem Satz vorangestellte „Hausadresse". Diese Angabe wird jeweils der elektronischen Vergleichsschaltung zugeführt.

der Obergrenze (Zylinder 013, Spur 4). Die Obergrenze ergibt sich aus der Addition der absoluten Bereichsgröße (4 Zylinder und 5 Spuren) zu der Untergrenze. Jetzt ließe sich natürlich auch leicht errechnen, wo bei einer fortlaufenden Speicherung beispielsweise der 805. Satz stehen müßte, nämlich im Zylinder 013 an der fünften Stelle der Spur 2. Die „Hausadresse" dieses 805. Satzes lautete also Zylinder 013, Spur 2, Satz-Nr. 4.
Um die Speicherkapazität der Magnetplatte gut auszunutzen, kann ebenso wie bei dem Magnetband der Begriff der Blockung Anwendung finden. Auf diese Art und Weise wird der Anteil der „Lücken" zwischen den einzelnen Sätzen klein gehalten. Tabellen, die von den Herstellerfirmen der Plattenstapel zu beziehen sind, geben darüber Auskunft, wieviel Datensätze sich mit oder ohne Blockung innerhalb einer Spur unterbringen lassen. Solche Angaben sind für den Programmierer sehr wichtig. Denn dieser muß nach Abschätzung der Gesamtzahl aller für einen Datenbestand anfallenden Sätze die dazugehörigen Bereichsgrenzen in der Dateibeschreibung festlegen.
Um ein vereinfachtes Beispiel zu geben, wie zum Beispiel eine Rechnungsschreibung mit Hilfe von Auftrags-Lochkarten, einer Kundenanschrifts- und einer Artikel-Datei bestritten werden kann, sei folgendes angenommen (Abb. 169):
Die Kundensätze – bestehend aus der Kundennummer, dem Kundenamen und der Anschrift – sind in dem Bereich vom Zylinder 020, Spur 0 bis zum Zylinder 100, Spur 9 gespeichert.



Schema einer Rechnungsschreibung

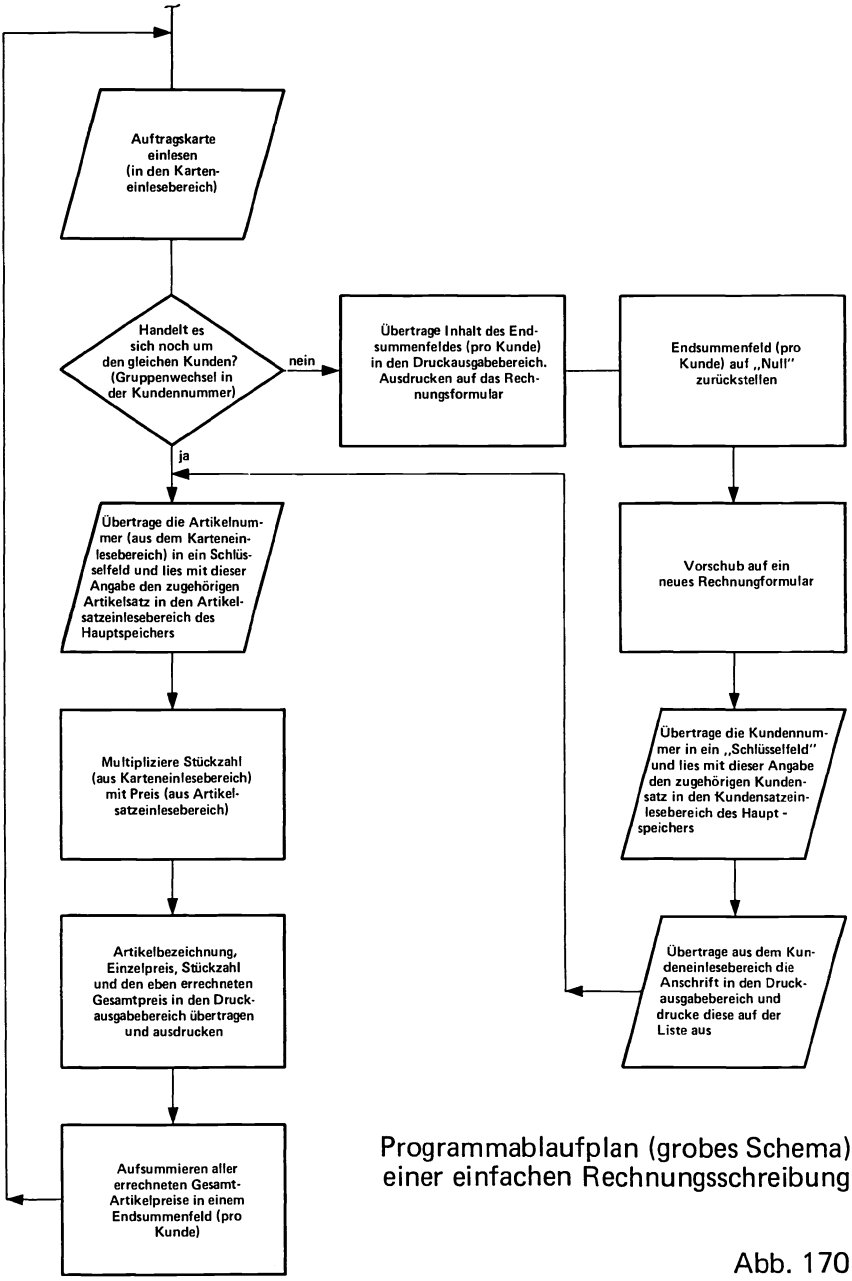
Auftrags-Karten enthalten: **Kundennummer, Artikelnummer, Stückzahl**

Kundensatz enthält: **Kundennummer, Kundenname, Kundenanschrift**

Artikelsatz enthält: **Artikelnummer, Artikelbezeichnung, Preis**

Abb. 169

Die Artikelsätze, eine Artikelnummer und einen Preis umfassend, belegen den Bereich vom Zylinder 101, Spur 0 bis zum Zylinder 110, Spur 9.
Die Auftrags-Karten enthalten jeweils eine Kundennummer, die Nummer und Stückzahl des bestellten Artikels. Sie werden nach der Kundennummer sortiert über den Karten-Leser der Programmverarbeitung zugeführt.



Programmablaufplan (grobes Schema) einer einfachen Rechnungsschreibung

Abb. 170

Liegen von einem Kunden mehrere Bestellungen vor, so sind auch entsprechend mehrere Auftrags-Karten vorhanden. Es wäre also denkbar, daß bei der Rechnungsschreibung auf fünf Auftrags-Karten mit der Kundennummer 0023 zehn Auftrags-Karten mit der Kundennummer 0031 usw. folgen. Für jeden Kunden soll ein Rechnungsformular erstellt werden.
Der Rechnungskopf (Abb. 148) umfaßt das Datum und die Kundenanschrift, der folgende Teil der Rechnung die verschiedenen Artikelpositionen und die Endsumme.
Nun sind nur noch einige Bemerkungen erforderlich, um das Blockdiagramm für die Rechnungsschreibung (Abb. 170) verständlich zu machen. Man ist vielleicht noch etwas von der sequentiellen Bandverarbeitung voreingenommen und vermutet, daß die Verarbeitung einer Platten-Datei in ähnlicher Art und Weise erfolgt. Diese Annahme findet sich jedoch nur zum Teil bestätigt. Denn für Plattensätze gibt es sowohl eine sequentielle (fortlaufende) als auch „gezielte" Verarbeitungsformen.

Die sequentielle Verarbeitungsform

Hierbei wird Satz für Satz nacheinander in den Hauptspeicher eingelesen und verarbeitet: (Eine Möglichkeit, von der man zum Beispiel beim Erstellen einer Kundenliste Gebrauch machen würde).

Die gezielten Verarbeitungsformen

Sie gestattet das gezielte Einlesen eines ganz bestimmten Datensatzes aus der Fülle einer Datei. Kein Wunder, daß so etwas möglich ist, schließlich kann der bewegliche S/L-Kopf durch einen entsprechenden Programmbefehl auf jeden beliebigen Punkt der Platte eingestellt werden.
Die Kennzeichnung des gewünschten Datensatzes erfolgt entweder durch Nennung seiner „Hausadresse" (Zylinder-, Spur- und Satz-Nummer) oder durch Angabe eines sogenannten „Schlüssels".
Von der ersten Möglichkeit wird dann Gebrauch gemacht, wenn von vornherein bekannt ist, unter welchen Hausadressen die gewünschten Sätze anzutreffen sind. Andernfalls müssen die Datensätze durch ihren „Schlüssel" bezeichnet werden.
Für Kundensätze wählt man zweckmäßigerweise die Kundennummer, für Artikelsätze die Artikelnummer als Schlüssel.
Wünscht man für eine vorliegende Kundennummer, zum Beispiel 876, den entsprechenden Kundensatz in den Kernspeicher einzulesen, so ist zweierlei erforderlich:

Übertragung der betreffenden Kundennummer in ein „Schlüsselfeld" des Kernspeichers
und der Lesebefehl GET KUNDE
└── symbolischer Dateiname für die Kundensätze

Alles Weitere erfolgt automatisch: Die Lokalisierung des gewünschten Kundensatzes auf der Platte und dessen Einlesen in den Kundensatz-Einlesebereich des Hauptspeichers.
(Für jede Platten-Datei sind ein eigener Einlesebereich und ein eigenes Schlüsselfeld im Hauptspeicher zu definieren.)
Jetzt ist es verhältnismäßig einfach, das „Grob"-Blockdiagramm für die Rechnungsschreibung in Abb. 170 zu verstehen:

Nach dem Einlesen einer Auftragskarte wird aufgrund der darin enthaltenen Kundennummer der zugehörige Kundensatz, und aufgrund der angegebenen Artikelnummer der zugehörige Artikelsatz eingelesen. Damit stehen aber alle Angaben für das Erstellen der Rechnung zur Verfügung und es ist nur noch eine Frage der Logik, mit dieser Information den Rechnungskopf bzw. die Artikelzeilen auszugeben.
Die vorstehenden, allgemein gehaltenen Bemerkungen zur Speicherung und Verarbeitung von Platten-Dateien werden im nächsten Kapitel näher und somit auch verständlicher erläutert.
Dies erscheint angebracht, da ohne ein solches Grundlagenwissen selbst allgemeine EDV-Kenntnisse nur lückenhaft wären.

22. Speicherungs- und Verarbeitungsformen von Magnetplatten-Dateien

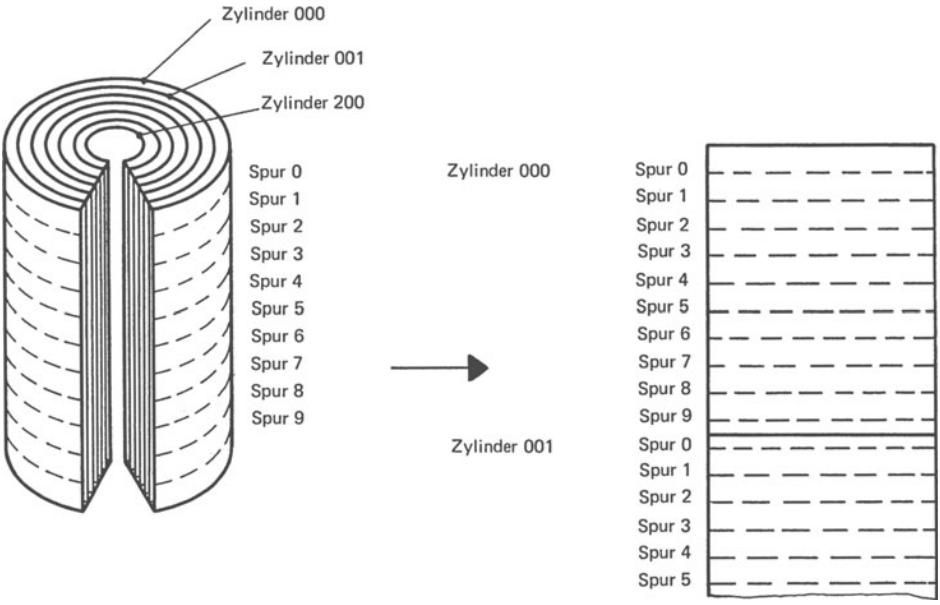
Es wurde bereits erwähnt, daß es auf Grund des beweglichen S-L-Kopfes möglich ist, die einzelnen Datensätze auf beliebige Stellen der Magnetplatte auszugeben.
Daraus resultieren verschiedene Speicherungsformen:

I. Die starr-fortlaufende Speicherungsform
II. Die gesteuerte Speicherungsform (direkte Adressierung)
III. Die logisch-fortlaufende Speicherungsform (mit Adreß-Index).

Auch das Einlesen der Datensätze kann nach unterschiedlichen Gesichtspunkten erfolgen, die die verschiedenen Verarbeitungsformen für Platten-Dateien bedingen:

Die starr-fortlaufende Verarbeitungsform
Die wahlweise Verarbeitungsform } „gezielte" Verarbeitungsformen.
Die logisch-fortlaufende Verarbeitungsform.

Für die folgenden Erklärungen ist vorteilhaft und übersichtlich, alle Plattenoberflächen eines Stapels schematisch als Rechteck darzustellen (Abb. 171).



Vereinfachte Darstellung aller Datenspuren in einem Rechteck

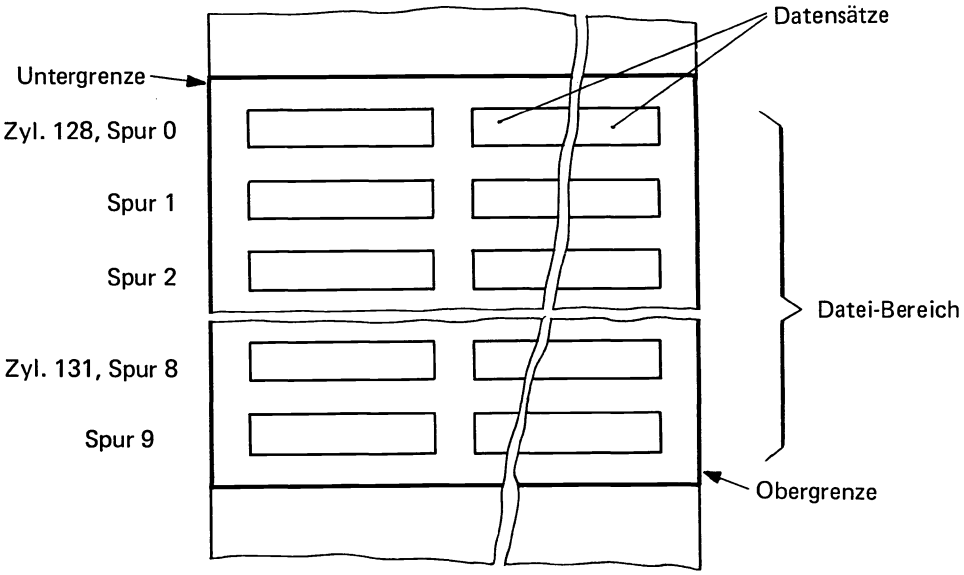
Abb. 171

Nach dem Zylinder-Konzept sind jeweils alle untereinanderliegenden Datenspuren auf einer gedachten Zylindermantelfläche angeordnet. Insgesamt gesehen, folgt daraus eine Aufteilung des Magnetplattenstapels in mehrere, ineinander liegende Zylinder mit den Datenspuren 0–9 (Abb. 166, 171).
In der vereinfachten Darstellung erscheinen diese Zylinder als untereinanderliegende Abschnitte, die kreisförmigen Datenspuren als gerade Linien (Abb. 171).
Wenn ein Magnetplattenstapel zur Datenspeicherung verwendet werden soll, ist jeder Datei (Artikelsätze, Kundensätze etc.) ein bestimmter Bereich zuzuweisen, in dem die Datensätze nach Maßgabe der gewählten Speicherungsform angeordnet werden können. Ein solcher Bereich ist durch Benennung seiner Untergrenze (z. B. Zylinder 128, Spur 0) und seiner Obergrenze (z. B. Zylinder 131, Spur 9) „abgesteckt" (Abb. 172). Solche Angaben sind in die Dateibeschreibung mit aufzunehmen.
Für die Ermittlung des Platzbedarfes einer Platten-Datei genügen drei Angaben:

1. Die Länge des Datensatzes
2. Der Hinweis, wie viele solcher Datensätze in einer Spur aufgenommen werden können.
3. Die maximale Anzahl der zu speichernden Datensätze.

$$\text{Gesamtzahl der benötigten Spuren} = \frac{\text{Maximale Anzahl der zu speichernden Datensätze}}{\text{Anzahl der Datensätze pro Spur}}$$

(1 Zylinder = 10 Spuren)



Abgrenzung eines Dateibereiches auf der Magnetplatte durch Angabe seiner Unter- und Obergrenze.

Abb. 172

Die Anordnung der Datensätze innerhalb des zugewiesenen Bereiches richtet sich nach der gewählten Speicherungsform. Sie wird durch die Erfordernisse einer wirkungsvollen Organisation festgelegt. Zur Erleichterung der „Rechenarbeit" stehen entsprechende Tabellen zur Ermittlung des Platzbedarfes der Platten-Dateien zur Verfügung. Sie sind von der Herstellerfirma des Computersystems zu beziehen.
Unter Organisation ist die programmtechnische Verknüpfung aller Arbeitsgebiete (Rechnungswesen, Buchhaltung, Lagerhaltung etc.) der elektronischen Datenverarbeitung in einem Betrieb zu verstehen.

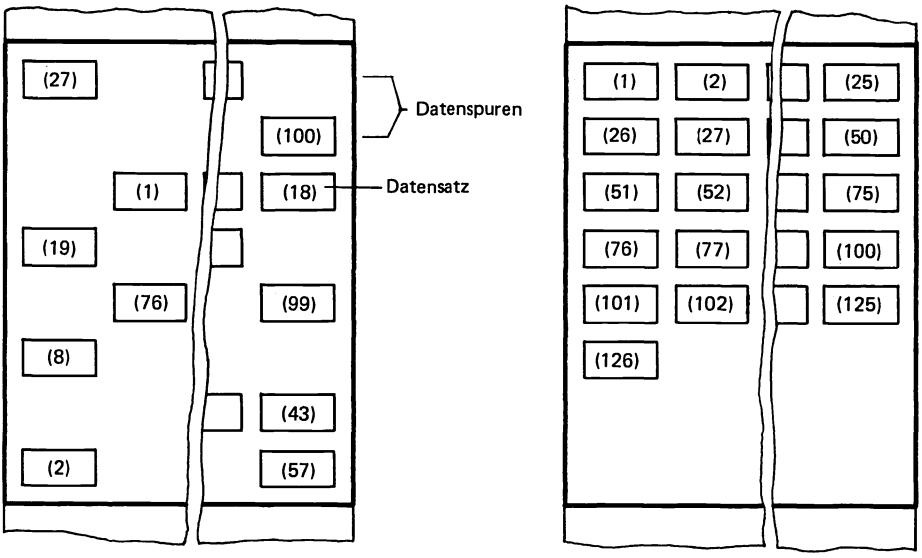
A. Speicherungsformen von Magnetplatten-Dateien

I. Die starr-fortlaufende Speicherungsform

Zur Ausgabe eines Datensatzes auf eine ausgewählte Stelle der Magnetplatte werden immer die Angaben zweier Hauptspeicherfelder,

des Plattenausgabebereiches und
des Plattenadreßfeldes

herangezogen. Der Inhalt des Plattenadreßfeldes, die „Hausadresse", bestimmt die künftige Position des Datensatzes auf dem Magnetplattenstapel.
Während des Programmlaufes wird für jeden auszugebenden Satz die „Hausadresse" nach besonderen Gesichtspunkten vorgegeben.
Die Anordnung der Sätze innerhalb ihres Dateibereiches kann recht unterschiedlich sein. Entweder erfolgt sie gestreut (Abb. 173a) oder nach einem starr-fortlaufenden Schema (Abb. 173b).



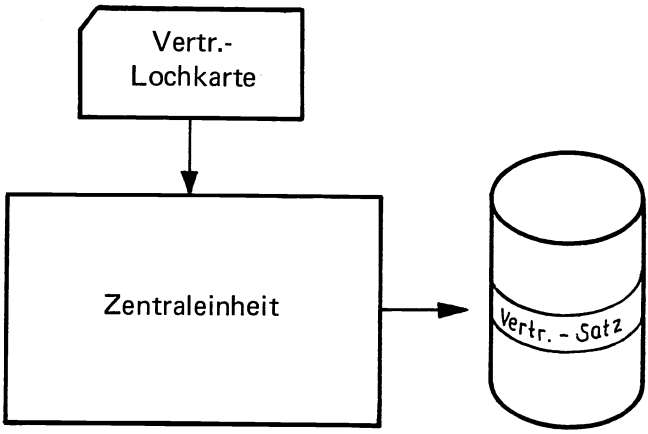
a) *gestreut* (mit Satzlücken) b) *starr-fortlaufend*

(die in Klammern gestellten Zahlen bedeuten die Reihenfolge, in der die Datensätze auf die Platte ausgegeben wurden)

Anordnung von Datensätzen auf der Magnetplatte

Abb. 173

Im letzteren Falle werden die Sätze in der Reihenfolge ihrer Ausgabe hintereinander (ohne Satzlücken) auf die Platte geschrieben. Beginnend bei der unteren Bereichsgrenze der Datei füllen sie Spur für Spur eines Zylinders.
Eine Datei dieser starr-fortlaufenden Speicherungsform wird später von den verschiedensten Programmen nach bestimmten Aspekten verarbeitet.
Um den Programmieraufwand für das Anlegen einer solchen Datei möglichst gering zu halten, sind spezielle Angaben in der Dateibeschreibung aufzunehmen. Diese Angaben machen die explizite Benutzung eines Plattenadreßfeldes im Programm überflüssig. Der Programmierer kann sich darauf beschränken, für die Datensätze lediglich einen Schreibbefehl zu erteilen. Dabei kann er sich darauf verlassen, daß sie alle durch Wirkung der Ein/Ausgabe-Steuerung automatisch auf den richtigen Platz geschrieben werden. Die Ein/Ausgabe-Steuerung enthebt ihn in diesem Falle der Arbeit, die für jeden Datensatz vorgesehene Hausadresse durch einen entsprechenden Befehl in das Plattenadreßfeld zu übertragen. (Der Anfangsinhalt des Plattenadreßfeldes müßte der unteren Datei-Bereichsgrenze entsprechen. Dieser Wert wäre nach jedem Schreibbefehl um 1 zu erhöhen, damit sich der nächste auszugebende Datensatz an den vorausgehenden starr-fortlaufend anschließen könnte. Die Ein/Ausgabe-Steuerung erledigt alle diese Schritte selbsttätig, ohne Zutun des Programmierers).
Wie schon so oft, soll auch hier ein Prinzip-Beispiel das Verständnis dieser Ausführungen erhärten. Es soll eine Platten-Datei der starr-fortlaufenden Speicherungsform mit Vertreter-Sätzen angelegt werden. Die Daten dazu entstammen Vertreter-Lochkarten (Abb. 174). Die Feldeinteilung der Vertreterlochkarten und Vertreter-Sätze ist in Abb. 175 angegeben. Die Länge des Datensatzes beträgt 80 Bytes.*



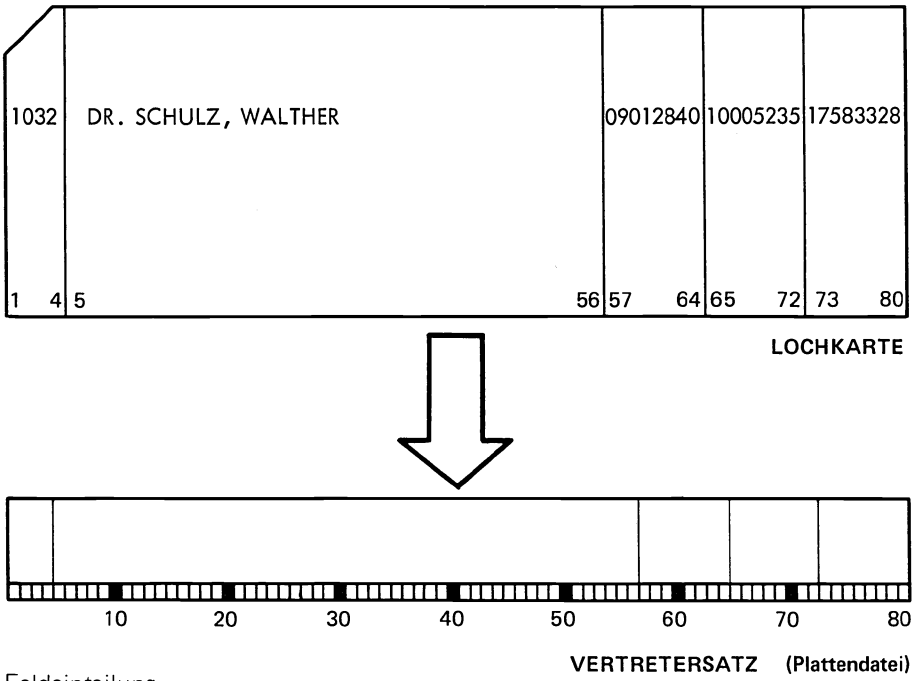
Platten-Einheit Nr. 2

Datenflußplan „ANLEGEN VON VERTRETER-SÄTZEN"

Abb. 174

* Die Plattensätze können je nach Festlegung auch eine kleinere bzw. größere Stellenzahl aufweisen. Ihr Inhalt muß nicht unbedingt aus Lochkarten, er kann ebensogut auch von einem Magnetband oder anderen Platten-Dateien etc. stammen.

Von einer Blockung sei aus Gründen der vereinfachten Darstellung abgesehen. Eine Abschätzung ergibt, daß maximal 1000 Lochkarten anfallen. Wenn in einer Spur bspws. 25 Sätze a 80 Bytes untergebracht werden können (laut Tabelle zur Ermittlung des Platzbedarfes), würde der benötigte Datei-Bereich demnach $^{1000}/_{25}$ = 40 Spuren (= 4 Zylinder) umfassen. Angenommen, der Magnetplattenstapel sei von fremden Datenbeständen bis zum Zylinder 127, Spur 9, bereits belegt. Folglich steht nur noch der verbleibende Teil zur freien Verfügung.
Davon werden jedoch nur $^{40}/_{10}$ = 4 Zylinder benötigt (1 Zylinder hat 10 Spuren). Wenn man den Zylinder 128, Spur 0, zur unteren Bereichsgrenze erklärt, liegt durch Addition von 4 Zylindern die obere Grenze bei Zylinder 131, Spur 9. Mit Hilfe dieser vorbereitenden Überlegungen kann jetzt endlich zur Dateibeschreibung des Programmes übergegangen werden (Abb. 176). Diese enthält die Speicherungsform der Datei als zusätzlichen Vermerk. Das Programmablaufschema ist in



Feldeinteilung

Lochkarte	Spalte	1 – 4	Vertreter-Nummer
	Spalte	5 – 56	Name
	Spalte	57 – 64	Umsatz 1968
	Spalte	65 – 72	Umsatz 1969
	Spalte	73 – 80	Umsatz 1970
Vertreter-Satz:	Stelle	1 – 4	Vertreter-Nummer
	Stelle	5 – 56	Name
	Stelle	57 – 64	Umsatz 1968
	Stelle	65 – 72	Umsatz 1969
	Stelle	73 – 80	Umsatz 1970

Abb. 175

Dateibeschreibungen

Für die Lochkarten auf der Kartenabfühleinheit

Name der Datei	KARTE
Art der Datei	INPUT
Name des Karteneinlesebereiches	KAB
EOF-Merkmal	SCHL

Für die Plattensätze des angegebenen Dateibereiches auf der Magnetplatten-Einheit 2

Name der Datei	PLAVER
Art der Datei	OUTPUT
Satzlänge	80
Name des Plattenausgabebereiches	PLAB 2
Untergrenze	ZYLINDER 128, SPUR 0
Obergrenze	ZYLINDER 131, SPUR 9
Speicherungsform	STARR-FORTLAUFEND

	OPEN	KARTE, PLAVER
ANF	GET	KARTE
	MVC	PLAB2, KAB
	PUT	PLAVER
	B	ANF
SCHL	CLOSE	KARTE, PLAVER
	EOJ	
**DEFINITIONEN		
KAB	DS	CL80
PLAB2	DS	CL80

Programm „Anlegen von VERTRETER-SÄTZEN"

Abb. 176

Abb. 177 wiedergegeben. Sobald der Schreibbefehl

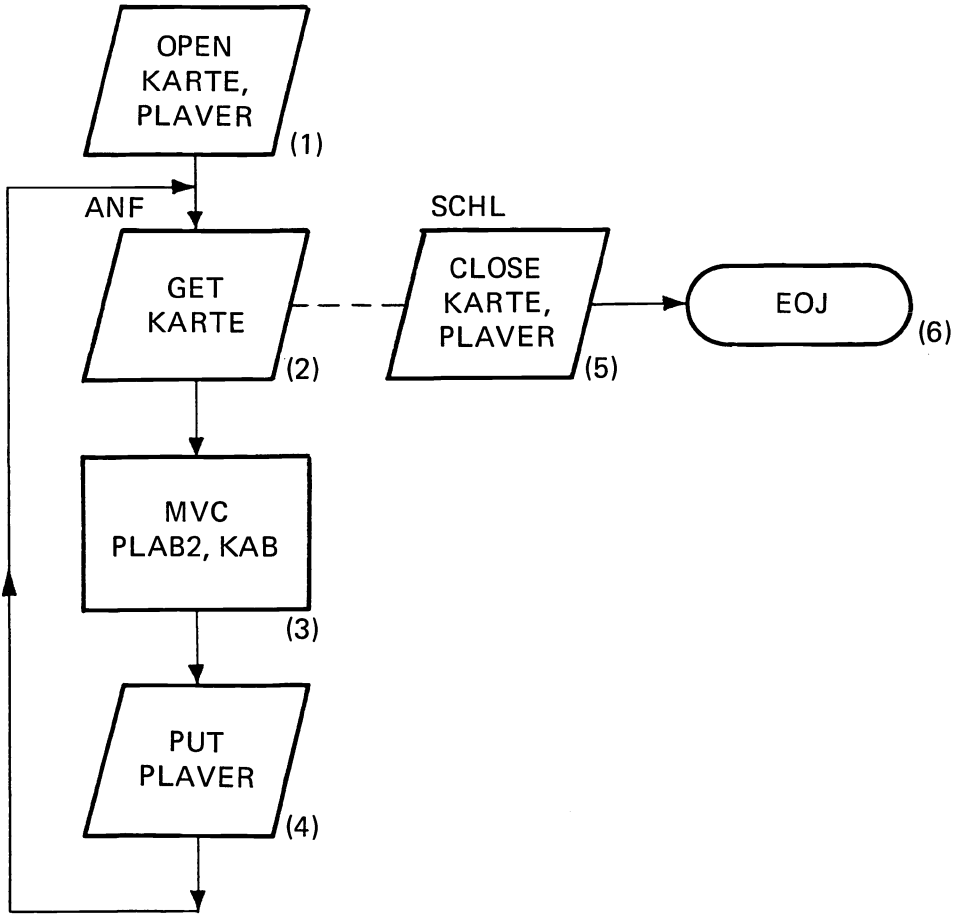
PUT PLAVER

für einen Plattensatz erstmals erteilt wird, erfolgt seine Ausgabe genau an der unteren Bereichsgrenze. Anschließend wird das Programm fortgeführt, bis ein neuer Satz (der zweite in diesem Programm) auszugeben ist. Dieser gelangt an eine Stelle der Platte, die sich an den vorhergehenden Satz anschließt (Abb. 178). Die Ausgabe der Vertretersätze auf die Platte wiederholt sich solange, bis das Karten-Ende erreicht ist. Zu diesem Zeitpunkt stehen bereits alle Datensätze in streng starr-fortlaufender Reihenfolge auf

der Platte (Abb. 178). Durch das Abschließen der Dateien mit

CLOSE KARTE, PLAVER

wird automatisch hinter den allerletzten Satz der Plattendatei PLAVER ein Datei-Ende-Zeichen „EOF" gesetzt. Es erlangt erst im Zusammenhang mit der Verarbeitung dieser Plattendatei durch ein anderes Programm an Bedeutung.



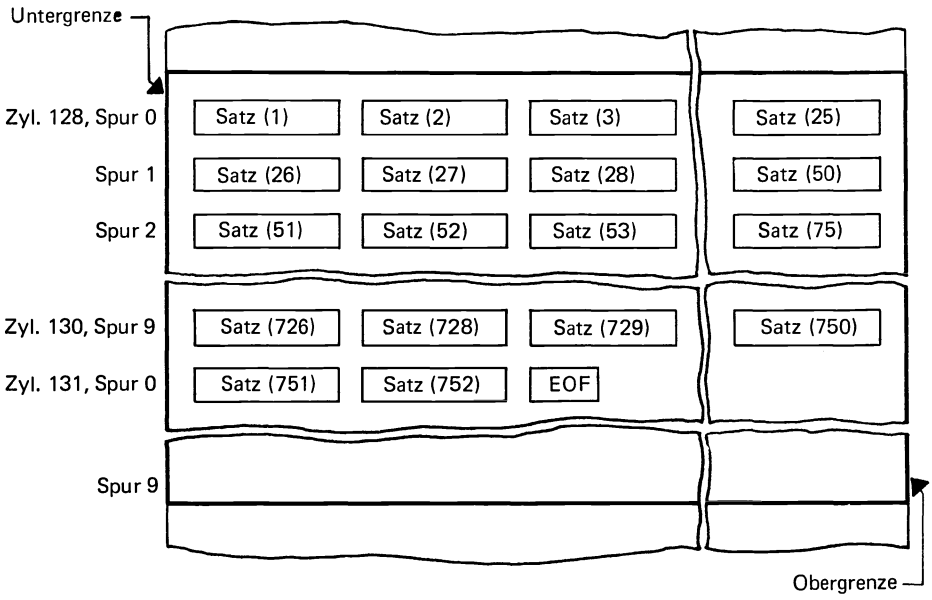
Zur Erläuterung:

(1) Eröffnen der Dateien
(2) Lochkarte nach KAB einlesen
(3) Übertragung des Inhalts von KAB (Karteneinlesebereich) nach PLAB2 (Plattenausgabebereich)
(4) Plattensatz schreiben

(5) Abschließen der Dateien
(6) Programmende (END OF JOB)

„ANLEGEN VON VERTRETERSÄTZEN"
Programmablaufplan (in ASSEMBLER-Instruktionen)

Abb. 177



Datei-Bereich starr-fortlaufend gespeicherter Sätze

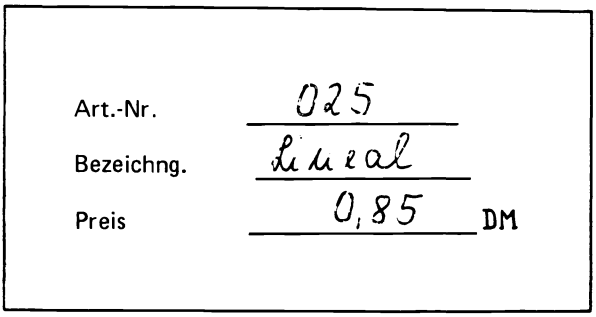
(Die in Klammern angegebenen Zahlen bedeuten die Reihenfolge, in der die Datensätze auf den Magnetplattenstapel geschrieben wurden. Auf die Angabe der einzelnen Hausadressen kann in dieser Darstellung verzichtet werden, da dem Programmierer ohnehin nur die obere und untere Bereichsgrenze bekannt sind).

Abb. 178

II. Die gestreute Speicherungsform

In jedem Handelsbereich gibt es eine Vielzahl von Artikeln. Jeder Artikel hat als Ordnungsbegriff eine Artikel-Nummer, der eine Artikelbezeichnung, ein Preis etc. zugeordnet sind. Der Umfang des Warensortiments bestimmt die Größe des Nummernkreises (z. B. Art.-Nr. 001 bis Art.-Nr. 400). Im Verlaufe der Zeit ergeben sich jedoch Änderungen:
Die Aufnahme neuer Artikel in das Warensortiment macht die Vergabe neuer Artikelnummern (bis 500) erforderlich. Mitunter werden auch Artikel aus dem Sortiment gestrichen – die Nummern dieser „Abgänge" bleiben frei. Außerdem sind Preisände-

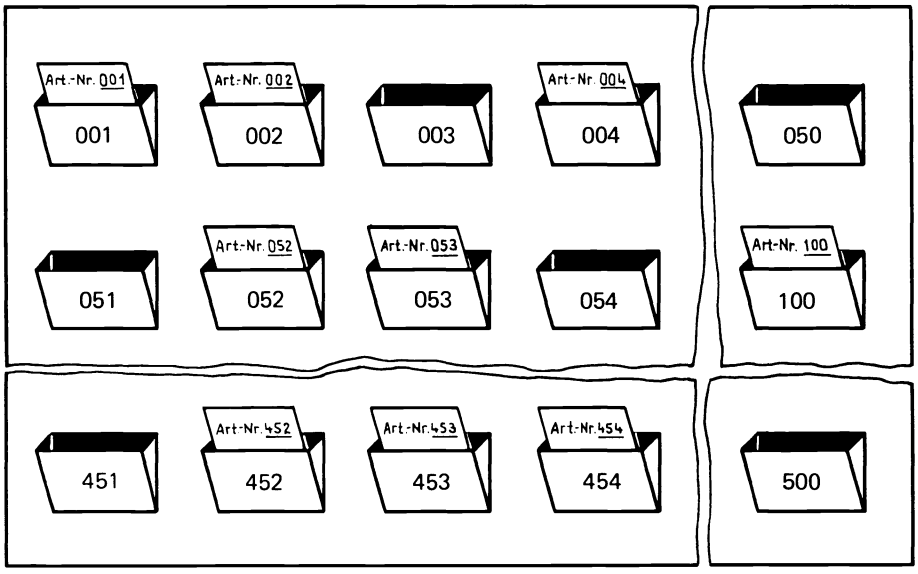
rungen nicht auszuschließen. Um einen ständigen Überblick zu haben, könnte man (theoretisch) auf die Idee kommen, sich eine Artikel-Kartei besonderer Art anzulegen. Für jeden Artikel wird eine Kartei-Karte (Abb. 179) ausgestellt.



Artikel-Kartei-Karte

Abb. 179

Jede dieser Karten soll in eines der fortlaufend durchnumerierten Fächer eines Kastens (Abb. 180) gesteckt werden. Es ist naheliegend, daß z. B. die Kartei-Karte mit der Artikel-Nummer 052 in das gleichermaßen bezeichnete Fach Nr. 052 gelangen soll. Dieser Einordnung liegt also ein bestimmtes Prinzip zugrunde: (in diesem Falle sind Art.-Nr. und Fachadresse gleich).



Artikel-Kartei mit „direktem Zugriff"

Abb. 180

Zylinder 132	Spur 0	132 0 00	(Artikelsatz mit der Art. Nr. 001)	132 1 01	(Artikelsatz mit der Art. Nr. 002)	132 0 24	(Artikelsatz mit der Art. Nr. 025)
	Spur 1	132 1 00	(Artikelsatz mit der Art. Nr. 026)	132 1 01		132 1 24	(Artikelsatz mit der Art. Nr. 050)
	Spur 2	132 2 00		132 1 01	(Artikelsatz mit der Art. Nr. 052)	132 2 24	
	Spur 3	132 3 00	(Artikelsatz mit der Art. Nr. 076)	132 1 01	(Artikelsatz mit der Art. Nr. 077)	132 3 24	(Artikelsatz mit der Art. Nr. 100)
	Spur 4	132 4 00	(Artikelsatz mit der Art. Nr. 101)	132 1 01		132 4 24	
	Spur 5	132 5 00	(Artikelsatz mit der Art. Nr. 126)	132 1 01	(Artikelsatz mit der Art. Nr. 127)	132 5 24	(Artikelsatz mit der Art. Nr. 150)
	Spur 6	132 6 00	(Artikelsatz mit der Art. Nr. 151)	132 1 01		132 6 24	(Artikelsatz mit der Art. Nr. 175)
	Spur 7	132 7 00	(Artikelsatz mit der Art. Nr. 176)	132 1 01		132 7 24	(Artikelsatz mit der Art. Nr. 200)
	Spur 8	132 8 00	(Artikelsatz mit der Art. Nr. 201)	132 1 01		132 8 24	
	Spur 9	132 9 00	(Artikelsatz mit der Art. Nr. 226)	132 1 01	(Artikelsatz mit der Art. Nr. 227)	132 9 24	(Artikelsatz mit der Art. Nr. 250)
Zylinder 133	Spur 0	133 0 00	(Artikelsatz mit der Art. Nr. 251)	133 1 01	(Artikelsatz mit der Art. Nr. 252)	133 0 24	
	Spur 1	133 1 00		133 1 01	(Artikelsatz mit der Art. Nr. 277)	133 1 24	(Artikelsatz mit der Art. Nr. 300)
	Spur 2	133 2 00		133 1 01	(Artikelsatz mit der Art. Nr. 302)	133 2 24	(Artikelsatz mit der Art. Nr. 325)
	Spur 3	133 3 00	(Artikelsatz mit der Art. Nr. 326)	133 1 01		133 3 24	(Artikelsatz mit der Art. Nr. 350)
	Spur 4	133 4 00	(Artikelsatz mit der Art. Nr. 351)	133 1 01	(Artikelsatz mit der Art. Nr. 352)	133 4 24	
	Spur 5	133 5 00		133 1 01	(Artikelsatz mit der Art. Nr. 377)	133 5 24	(Artikelsatz mit der Art. Nr. 400)
	Spur 6	133 6 00	(Artikelsatz mit der Art. Nr. 401)	133 1 01		133 6 24	(Artikelsatz mit der Art. Nr. 425)
	Spur 7	133 7 00	(Artikelsatz mit der Art. Nr. 426)	133 1 01	(Artikelsatz mit der Art. Nr. 427)	133 7 24	
	Spur 8	133 8 00		133 1 01	(Artikelsatz mit der Art. Nr. 452)	133 8 24	(Artikelsatz mit der Art. Nr. 475)
	Spur 9	133 9 00	(Artikelsatz mit der Art. Nr. 476)	133 1 01		133 9 24	(Artikelsatz mit der Art. Nr. 500)

Plattenbereich für Artikel-Sätze

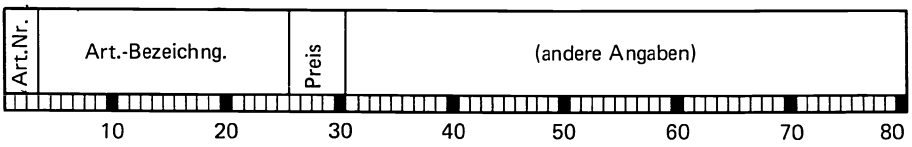
Abb. 181

Jedem Artikel ist auf Grund seines Ordnungsbegriffes eine ganz bestimmte Fachadresse zugewiesen. Die Reihenfolge, in der die „Einsortierung" der Kartei-Karten erfolgt, ist somit belanglos. Alle Fächer, für die es keine Kartei-Karten gibt, bleiben leer. In diesem Zusammenhang wäre man bereits berechtigt, von einer gestreuten Speicherung zu sprechen. Der Begriff „gestreut" ist deshalb so markant, weil die Kartei-Karten über den gesamten Kasten ***verstreut*** in den ihnen zugeordneten Fächern verteilt sind (Abb. 180).*) Die Zuordnung eines bestimmten Faches für einen bestimmten Artikel erscheint recht sinnvoll. So kann man in kürzester Zeit Auskunft über einen bestimmten Artikel erhalten – ein Griff in das entsprechende Fach und schon ist die Kartei-Karte mit allen wichtigen Angaben zur Hand.
Wie man bei Abgängen, Zugängen bzw. bei Preisänderungen verfahren würde, bedarf wohl keiner ausdrücklichen Erwähnung mehr.
Dieses Beispiel diente lediglich dazu, um auf die gestreute Speicherungsform von Artikelsätzen auf der Magnetplatte überzuleiten. Man braucht sich dabei lediglich den Karteikasten (Abb. 180) durch einen Plattenbereich (Abb. 181) und die Kartei-Karten durch Lochkarten (Abb. 182) ersetzt zu denken. Der Inhalt jeder Lochkarte soll als Artikelsatz (Abb. 183) auf eine ausgewählte Stelle der Magnetplatte geschrieben werden. Diese Stelle entspricht im übertragenen Sinne einer bestimmten Fachadresse des Kartei-Kastens. Sie ist durch Angabe der Plattenadresse nach Zylinder-, Spur- und Satznummer zu kennzeichnen. Der Programmierer würde bei einer solchen Aufgabe wie folgt vorgehen: Entsprechend dem Artikel-Nummernkreis 001 bis 500 sind 500 durchlaufend numerierte Plätze auf der Magnetplatte zu reservieren. Ein solcher Platz muß so groß sein, daß er einen Artikelsatz aufnehmen kann. Die Angaben des Artikelsatzes sollen aus einer Artikel-Lochkarte stammen.

Feldeinteilung einer Artikel-Lochkarte

Abb. 182

* Eine starr-fortlaufende Speicherung der Kartei-Karten ergäbe sich dann, wenn diese ohne Rücksicht auf Art.-Nr. und Fachadresse unmittelbar aufeinanderfolgend in die Fächer dieses Kastens gesteckt worden wären.



Feldeinteilung eines Artikel-Satzes

Abb. 183

In einer Spur haben 25 dieser Sätze (à 80 Bytes) Platz, so daß der benötigte Dateibereich

$$\frac{500}{25} = 20 \text{ Spuren} = 2 \text{ Zylinder}$$

umfaßt.
Allerdings sind die 500 Plätze nicht im Sinne von 1,2,3 . . . bis 500 durchnumeriert; sie unterliegen vielmehr einer Kennzeichnung nach Zylinder-, Spur- und Satznummer.
Zu ihrer genauen (absoluten) Positionsangabe sind zunächst die Datei-Bereichsgrenzen festzulegen. Dabei ist man an den noch frei verfügbaren Raum des Plattenstapels gebunden.

Artikelsätze	Untergrenze	Zyl. 132	Spur 0
Artikelsätze	Obergrenze	Zyl. 133	Spur 9

(Abb. 181, 203). Ein Datensatz für den Artikel Nr. 298 wäre demnach an dem 298ten Platz der Datei zu speichern; ungerechnet entspräche dies dem Zylinder 133, Spur 1, Satznr. 22 (Abb. 181).
Die Umrechnung von der Art.-Nr. auf die Plattenadresse erfolgt nach einem recht einfachen Prinzip:

Man geht davon aus, daß

in einer Spur	(z. B. Spur 0)	25 Sätze
in zwei Spuren	(z. B. Spur 0 und 1)	50 Sätze
in drei Spuren	(z. B. Spur 0, 1 u. 2)	75 Sätze
	.	
	.	
	.	
in zwanzig Spuren	(z. B. Spur 0 des einen bis zur Spur 9 des nächsten Zylinders	500 Sätze

untergebracht sein können. Innerhalb einer Spur sind die Satz-Nr. 0 bis 24 vergeben (insgesamt also 25 Sätze).
Um für die Art.-Nr. 298 die Plattenadresse Zyl. 133, Spur 1, Satz-Nr. 22 zu erhalten, genügen drei Rechenschritte.

1. $\frac{\text{Art.-Nr.}}{\text{Anzahl der Sätze pro Spur}} = \frac{298}{25} = \underline{1} \quad \underset{\sim}{1}$ Rest 23

2. Nach Verminderung des Restes um 1 verbleiben $\underline{1} \quad \underset{\sim}{1}$ Rest 22 (neu)

Es bedeuten:

- die links neben der ganzzahligen Einerstelle stehende Zahl 1 (unterstrichen) die relative Zylinder-Nr. 1
- die ganzzahlige Einerstelle 1 (durch Wellenlinie gekennzeichnet) die relative Spur-Nr. 1
- der Rest 22 (neu) die relative Satznummer.
- Die relative Satzadresse Zyl. 1, Spur 1, Satz-Nr. 22 kennzeichnet die Lage des Datensatzes für die Art.-Nr. 298, wenn die untere Bereichsgrenze bei Zyl. 000, Spur 0, läge. Zur Bestimmung der wahren (absoluten) Satzadresse ist das errechnete Ergebnis zu der festgelegten Untergrenze (Zyl. 132, Spur 0) zu addieren.

3. Addition der weiteren Bereichsgrenze Zyl. 132, Spur 0, zu dem unter 2. errechneten Ergebnis

Zyl. 1, Spur 1, Satz-Nr. 22
Zyl. 132, Spur 0
Zyl. 133, Spur 1, Satz-Nr. 22

Diese Umrechnungsmethode gilt für alle Artikelnummern. Sie ist eine der wichtigsten Voraussetzungen für das Programm „ARTIKEL-SÄTZE ANLEGEN" (Abb. 184).

Hinweis zur Abb. 184: Die Blöcke (1) bis (4) stellen eine Zusammenfassung mehrerer Assembler-Instruktionen dar, auf deren ausführliche Erläuterung in diesem Zusammenhang verzichtet wird.
Die programmierte Umrechnungsroutine (Artikelnummer in Plattenadresse) beschränkt sich lediglich auf das Prinzip und erhebt keinen Anspruch auf Exaktheit bis ins Detail. Auf die Definitionsangaben wurde verzichtet.

Nach dem Einlesen einer Lochkarte wird deren Artikelnummer in eine entsprechende Plattenadresse umgerechnet. Dieser Wert ist in das Plattenadreßfeld PLAADR zu übertragen. Der Name dieses Hauptspeicherfeldes ist in der Dateibeschreibung festgelegt, seine Größe unter Definitionen vereinbart.
Der jeweilige Inhalt des Plattenadreßfeldes bestimmt die Einstellung des S-L-Kopfes in die gewünschte Position, so daß nach Erteilung des Schreibbefehles PUT PLARTI der Artikelsatz (aus dem Plattenausgabebereich PLABER für die Datei PLARTI) auf die errechnete Stelle der Platte geschrieben wird.
Wurden innerhalb des Nummernkreises (001—500) bestimmte Artikelnummern nicht vergeben, so können dafür auch keine Artikelsätze angelegt werden. Die ihnen zustehenden Plätze auf der Platte bleiben frei. Duruch derartige „Lücken" bleibt die Belegungsdichte der Artikel-Datei kleiner als 100 % (Abb. 181).

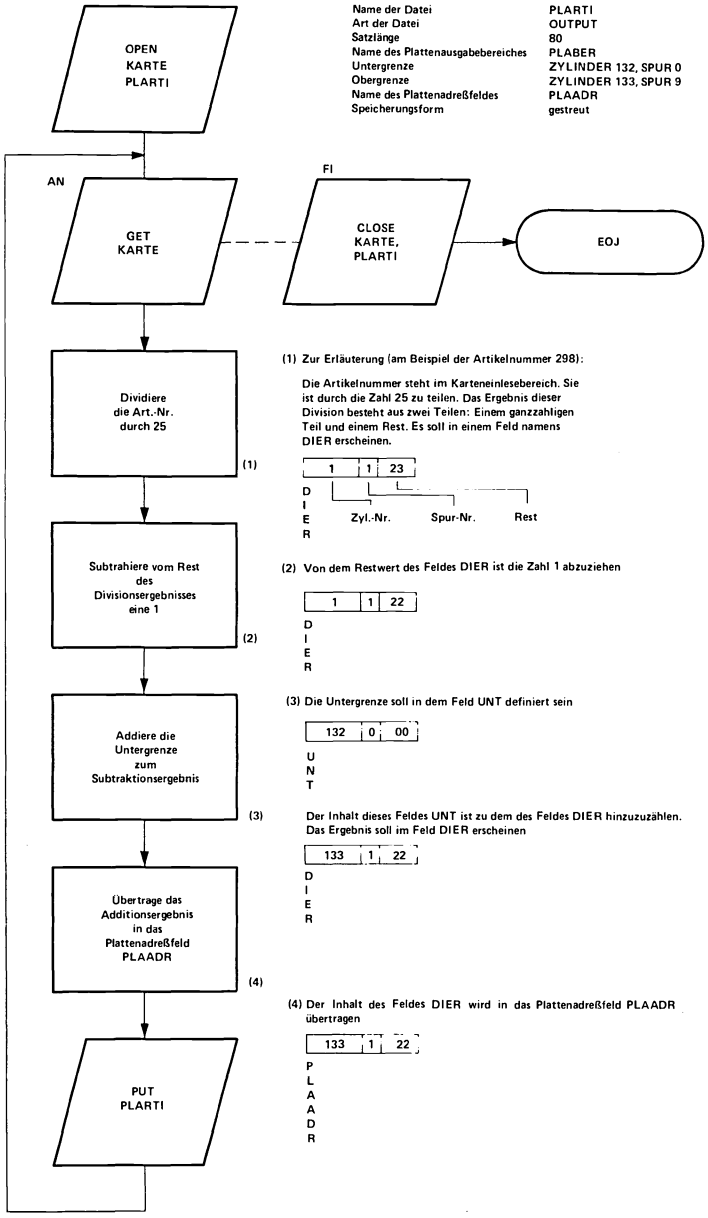
Dateibeschreibungen

Für die Lochkarten auf der Kartenabfühleinheit

Name der Datei	KARTE
Art der Datei	INPUT
Name des Karteneinlesebereiches	KABER
EOF-Merkmal	

Für die Plattensätze des angegebenen Dateibereiches auf der Magnetplatteneinheit 2

Name der Datei	PLARTI
Art der Datei	OUTPUT
Satzlänge	80
Name des Plattenausgabebereiches	PLABER
Untergrenze	ZYLINDER 132, SPUR 0
Obergrenze	ZYLINDER 133, SPUR 9
Name des Plattenadreßfeldes	PLAADR
Speicherungsform	gestreut

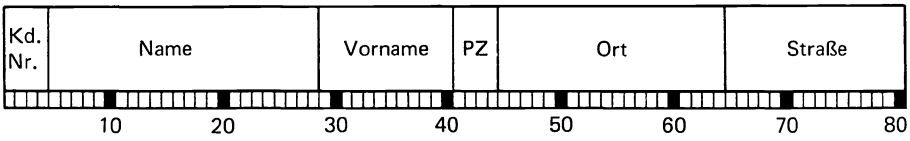


Programm „ARTIKEL-SÄTZE ANLEGEN"

Abb. 184

III. Die logisch-fortlaufende Speicherungsform

Die Zweckmäßigkeit dieser speziellen Speicherungsform soll wieder anhand eines Beispiels aufgezeigt werden.
Es sind insgesamt 4000 Kundensätze (Abb. 185), deren Angaben aus Lochkarten (Abb. 186) stammen, auf der Platte zu speichern. Die Kundennummer verteilt sich (streut) über den Bereich von 0001 bis 9999. Von den insgesamt 9999 möglichen Nummer sind lediglich 40000 belegt. Die Belegungsdichte beträgt demnach 4000/9999 = 40 %.



Feldeinteilung eines Kunden-Satzes

Abb. 185

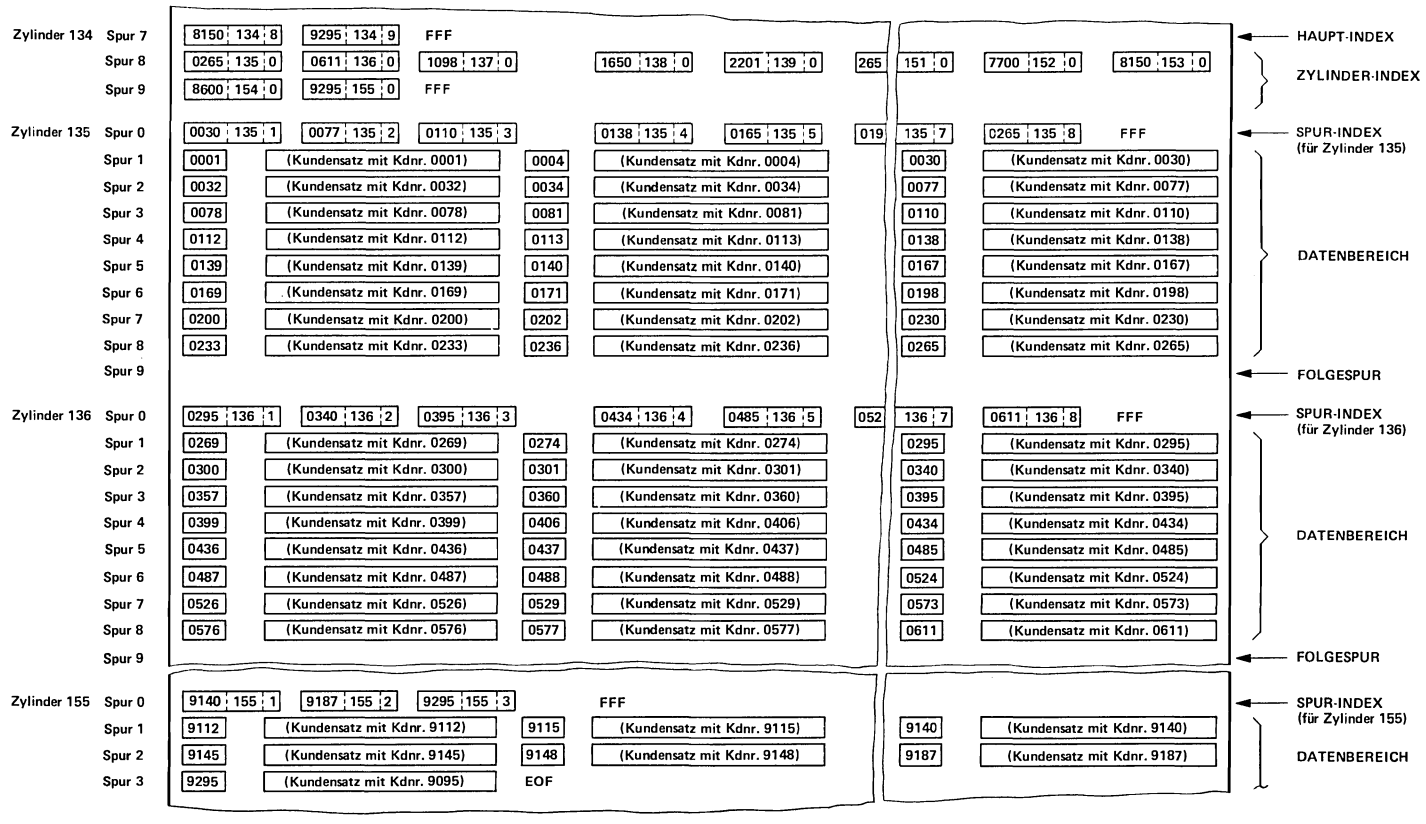
Abb. 186

Nach den vorangegangenen Ausführungen bieten sich zum Anlegen einer Kunden-Datei zwei Möglichkeiten: Die starr-fortlaufende oder die gestreute Speicherungsform.
Im ersteren Falle hätte der Dateibereich 4000 Sätze à 80 Bytes zu umfassen. Der Platzbedarf beträgt bei 25 Sätzen / Spur 4000/25 = 160 Spuren = 16 Zylinder. Sämtliche Datensätze wären ab Untergrenze streng nacheinander in der Reihenfolge der eingegebenen Lochkarten zu speichern. Der Operator führte diese Lochkarten vor Beginn des Programmlaufes als Kartenpaket dem Karten-Leser zu. Er schaut sie sich dabei nicht im einzelnen an, um eventuell herauszufinden, daß bspws. in der 73. Lochkarte die Kundennummer 4365 abgelocht ist. Nach beendetem Programmlauf wäre also nicht bekannt, an welcher Stelle der Platte der Datensatz mit der Kundennummer 4365 anzutreffen ist. Der Fachmann bezeichnet diesen Tatbestand folgendermaßen: Zwischen dem Ordnungsbegriff und der Satzadresse besteht kein rechnerischer Zusammenhang. (Ein solcher läßt sich nur bei der gestreuten Speicherungsform herstellen).

Würde man für die Kunden-Datei die gestreute Speicherungsform wählen, so müßte das Programm zum Anlegen der Kundensätze eine Umrechnungsroutine (Kundennummer → Satzadresse) vorsehen. Auf der Platte ist für jede theoretisch mögliche Kundennummer 0001—9999) ein entsprechender Platz zu reservieren, an dem der zugehörige Datensatz gespeichert werden kann. Bei geringer Belegungsdichte bleiben viele dieser Plätze frei, was letztlich zu einer unvollständigen Kapazitätsausnutzung des Datei-Bereiches führt. Während nach der starr-fortlaufenden Speicherungsform für die Kunden-Datei nur 16 Zylinder benötigt werden, erfordert die gestreute Speicherungsform 40 Zylinder (für jede Kundennummer von 0001 bis 9999 ist ein entsprechender Platz à 80 Bytes auf der Platte zu reservieren. Daraus folgt ein Platzbedarf von 9999/25 = 400 Spuren = 40 Zylinder. In diesem Bereich wären die anfallenden 4000 Datensätze gestreut zu speichern). Auf Grund der Umrechnungsroutine besteht die Möglichkeit, zwischen Kundennummer und korrespondierender Satzadresse einen rechnerischen Zusammenhang herzustellen.
Die eine Speicherungsform (starr-fortlaufend) ist somit durch gute Ausnutzung der Speicherkapazität, die andere (gestreut) durch eine direkte Adressiermöglichkeit ausgezeichnet. Man kann das eine gegenüber dem anderen nicht als Vor- oder Nachteil werten. Jede Speicherungsform ist gemessen an der Funktion, die sie zu erfüllen hat, für sich die zweckmäßigste.
Allerdings steht die gestreute Speicherungsform bei sehr geringer Belegungsdichte dem Interesse an einer guten Kapazitätsausnutzung entgegen.
Dieser Umstand hat zur Entwicklung der logisch-fortlaufenden Speicherungsform (mit Adreß-Index) geführt.
Danach ist es möglich, die Sätze quasi unmittelbar nacheinander auf der Platte zu speichern, wobei zwischen der Plattenadresse jedes Datensatzes und seinem Ordnungsbegriff, der Kundennummer, ein Zusammenhang (kein rechnerischer Zusammenhang!) besteht.
Dieser Zusammenhang wird von dem sogenannten Adreß-Index hergestellt. Von ihm leitet sich der Begriff des „indirekten Zugriffes" ab. Die Erläuterungen dazu fallen um so verständlicher aus, wenn man von einem bereits logisch-fortlaufend gespeicherten Datenbestand ausgeht (Abb. 187): Jedem Datensatz auf der Platte ist neben der üblichen Satzadresse ein „Schlüssel" vorangestellt (Abb. 188). Er enthält den Ordnungsbegriff, die Kundennummer. Sämtliche Datensätze sind in der Reihenfolge aufsteigender Kundennummern gespeichert.
Die Verarbeitung einer solchen Datei setzt das Einlesen jeweils eines Satzes in den Hauptspeicher voraus. Je nach Anforderung kann dies einmal der Kundensatz mit dem Schlüssel 0406, ein anderes Mal mit dem Schlüssel 9145 usw. sein. Dieser Hinweis deutet bereits an, daß ein Lesebefehl immer für einen Datensatz mit einer ganz bestimmten Kundennummer wirksam wird. Die gewünschte Kundennummer ist im Programm in einem sogenannten Schlüsselfeld des Hauptspeichers bereitzustellen (Abb. 189). Sein Name (z. B. SCHLUE) wird in der Dateibeschreibung festgelegt, seine Größe unter Definitionen vereinbart (Abb. 191).
Allein diese Angabe genügt, um den so bezeichneten Datensatz einlesen zu lassen.
Es wird angenommen, daß das Schlüsselfeld SCHLUE einen Inhalt von 0301 hat. Nach Ausführung des Lesebefehles GET KUND soll der Kundensatz mit der Kundennummer 0301 im Platteneinlesebereich der Datei KUND stehen. Dabei spielen sich folgende Vorgänge ab:
Der S-L-Kopf stellt sich zunächst auf die Spur (Abb. 187) ein, die den sogenannten

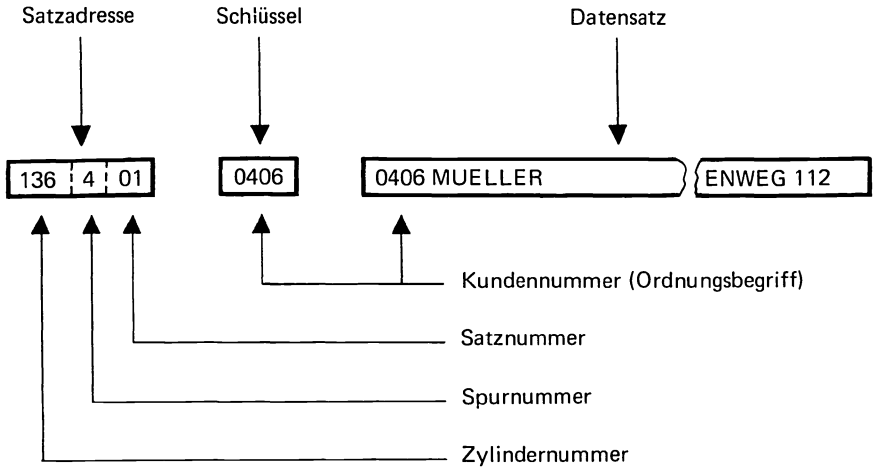
Haupt-Index

enthält.



Beispiel eines logisch-fortlaufend gespeicherten Datenbestandes

Abb. 187



Beispiel eines logisch-fortlaufend gespeicherten Datensatzes mit Satzadresse und Schlüssel

Abb. 188

Er enthält Angaben, die jeweils eine Kundennummer und eine Plattenadresse umfassen (Abb. 190).
Da die Platte ständig rotiert (ab Spurbeginn), werden alle Hauptindizes nacheinander vom S-L-Kopf gelesen und elektronisch ausgewertet. (Anmerkung auf Seite 267). Dabei findet ein elektronischer Vergleich zwischen dem Inhalt des Hauptspeicherfeldes SCHLUE und der soeben gelesenen Kundennummer der Hauptindexangabe statt. Das Vergleichsergebnis lautet entweder

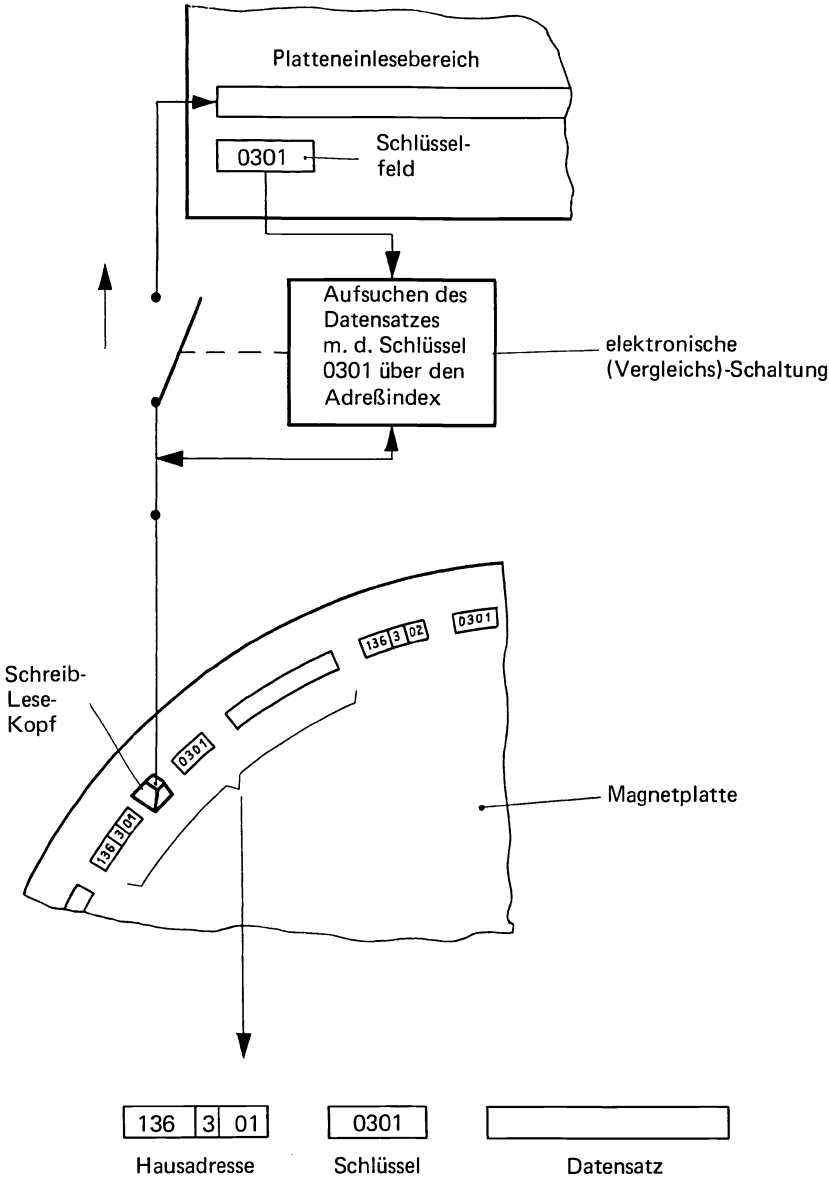
a) Die gewünschte Kundennummer im Schlüsselfeld SCHLUE ist kleiner oder gleich der Kundennummer in der Hauptindexangabe;
b) Die gewünschte Kundennummer im Schlüsselfeld SCHLUE ist größer als die Kundennummer in der Hauptindexangabe.

Im Beispiel nach Abb. 189 trifft das Ergebnis a) zu; die hinter der Kundennummer 8150 der Hauptindexangabe stehende Plattenadresse Zylinder 134, Spur 8 des Zylinderindex wird zur erneuten Einstellung des S-L-Kopfes benutzt. (Andernfalls Ergebnis b) – würde die nächste Hauptindexangabe gelesen und wie beschrieben ausgewertet werden).

Der

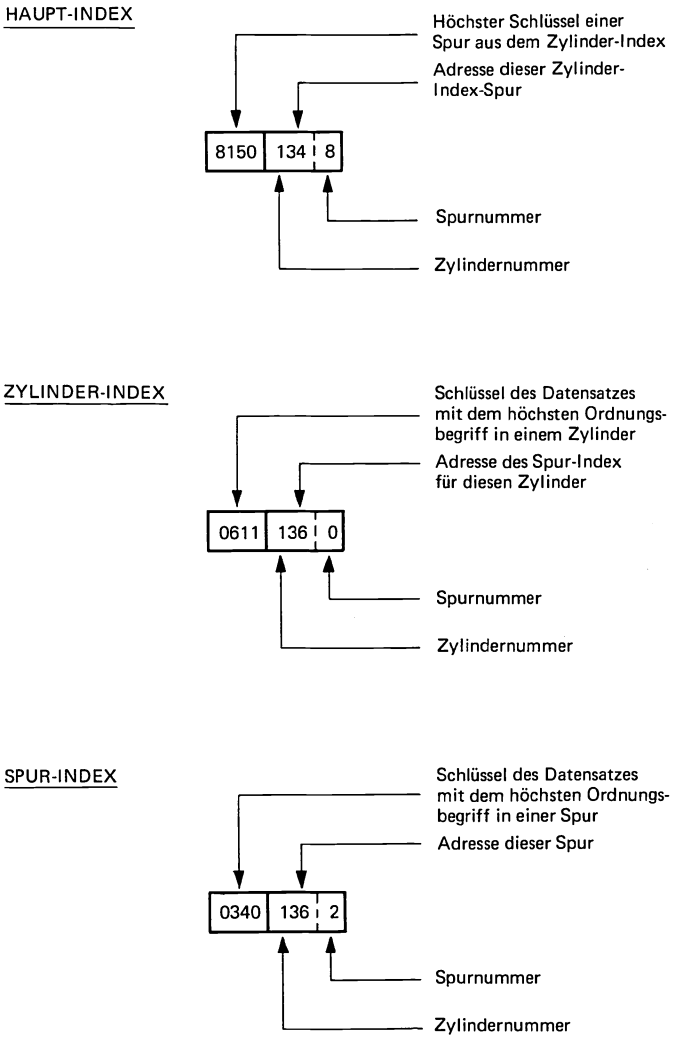
Zylinder-Index

(Abb. 187) enthält wiederum Angaben, die jeweils eine Kundennummer und eine Plattenadresse umfassen (Abb. 190).



Einlesen eines logisch-fortlaufend gespeicherten Datensatzes durch Bereitstellen seines Ordnungsbegriffes (z. B. der Kundennummer) im Schlüsselfeld des Hauptspeichers

Abb. 189



Angaben zum Adreß-Index eines logisch fortlaufend gespeicherten Datenbestandes

(Die angegebenen Adressen beziehen sich auf das Aufsuchen des Kundensatzes mit der Kdnr. 0301 in der Kunden-Datei lt. Abb. 187).

Abb. 190

Der S-L-Kopf ist jetzt auf Zylinder 134, Spur 8 des Zylinderindex eingestellt. Die Zylinderindexangaben wurden (ab Spurbeginn) nacheinander gelesen und elektronisch ausgewertet. Der elektronische Vergleich der gewünschten Kundennummer im Schlüsselfeld SCHLUE mit der Kundennummer der soeben gelesenen Zylinderindexangabe führt zu einem der beiden Ergebnisse c) oder d):

c) Die gewünschte Kundennummer im Schlüsselfeld SCHLUE ist kleiner bzw. gleich der Kundennummer in der Zylinderindexangabe.
d) Die gewünschte Kundennummer im Schlüsselfeld SCHLUE ist größer als die Kundennummer in der Zylinderindexangabe.

Fällt die Entscheidung zugunsten des Ergebnisses c), so wird die hinter der Kundennummer der Zylinderindexangabe erscheinende Plattenadresse zur weiteren Einstellung des S-L-Kopfes benutzt. Im anderen Falle wird die nächste Zylinderindexangabe gelesen und wie beschrieben ausgewertet. Demnach stellt sich schließlich der S-L-Kopf auf Zylinder 136, Spur 8 ein.
Dort ist der sogenannte

Spur-Index

(Abb. 187) enthalten, der auf die Datensätze des gleichen Zylinders verweist. Der Spur-Index (Abb. 190) besteht aus Angaben, die sich jeweils aus einer Kundennummer und einer nachgestellten Plattenadresse zusammensetzen. Diese Angaben werden ab Spurbeginn nacheinander vom S-L-Kopf gelesen und elektronisch ausgewertet. Wie schon in den vorangegangenen Fällen, wird auch hier die geforderte Kundennummer im Schlüsselfeld SCHLUE mit der soeben eingelesenen Kundennummer der Spurindexangabe auf kleiner/gleich oder größer verglichen. Das Ergebnis lautet entweder:

e) Die gewünschte Kundennummer im Schlüsselfeld SCHLUE ist kleiner bzw. gleich der Kundennummer in der Spurindexangabe, oder
f) die gewünschte Kundennummer im Schlüsselfeld SCHLUE ist größer als die Kundennummer in der Spurindexangabe.

Im Falle e) wird die Satzadresse hinter der Kundennummer in der Spurindexangabe zur weiteren Einstellung des S-L-Kopfes benutzt, andernfalls ist die nächste Spurindexangabe zu lesen und wie beschrieben auszuwerten.
Am Beispiel der Kundennummer 0301 wird der S-L-Kopf bei der zweiten Spurindexangabe auf die Spur 2 des Zylinders 136 verwiesen. Dort stehen verschiedene Datensätze; unter ihnen auch der gewünschte. Jedem von ihnen ist der Schlüssel vorangestellt. Er wird gelesen und elektronisch im Schlüsselfeld SCHLUE mit der geforderten Kundennummer auf Gleichheit geprüft. Stimmen beide Werte überein, so darf man sicher sein, daß sich im Anschluß an den Schlüssel der gesuchte Kundensatz unter dem S-L-Kopf hindurchbewegt. Ihm wird daraufhin der Weg zum Einlesen in den Platteneinlesebereich des Hauptspeichers freigegeben.
Damit ist die Zweckmäßigkeit des Adreß-Index nachgewiesen: Der gewünschte Kundensatz mit der Kundennummer 0301 steht jetzt im Hauptspeicher. Zu seinem Aufsuchen genügte die Angabe seines Ordnungsbegriffes; (seine Plattenadresse kann vom Programmierer ohnehin nicht errechnet werden). Alles andere – die Orientierung im Adreß-Index – geschah automatisch unter der Ein/Ausgabe-Steuerung.

Nach vier Einstellbewegungen des S-L-Kopfes (auf den Haupt-, Zylinder-, Spur-Index und die ausgewiesene Datenspur) wurde der gewünschte Kundensatz auf der Magnetplatte lokalisiert.
Die Magnetplatte rotiert zwar rasend schnell; eine Adreß-Index-Angabe bewegt sich in Bruchteilen einer Tausendstel Sekunde am Schreib-Lesekopf vorbei. Sie wird in dieser kurzen Zeit gelesen und schlagartig ausgewertet (elektronischer Vergleich), so daß unmittelbar darauf die folgende Indexangabe ebenfalls sofort gelesen und ausgewertet werden kann. Während einer Plattendrehung werden somit alle Indexangaben in einer Spur erfaßt.*
Nach dem gleichen Prinzip, wie hier für die Kundennummer 0301 gezeigt wurde, kann auch jeder andere Plattensatz aufgesucht werden. Die geschickte Verknüpfung von Haupt-, Zylinder- und Spur-Index gestattet ein zielgerichtetes Einstellen des Schreib-Lese-Kopfes.
Umfaßt der Datenbestand einen sehr großen Plattenbereich, so können die anfallenden Zylinder-Index-Angaben mehrere Spuren füllen. Der Hauptindex verweist auf jeweils eine dieser Spuren des Zylinder-Index.
Der Spurindex steht immer am Anfang eines Zylinders in der Spur 0; die verbleibenden Spuren 1 bis 9 dienen der Aufnahme von Datensätzen. Erstreckt sich die Datei über mehrere Zylinder, so ist jedem von ihnen ein Spurindex in der Spur 0 vorangestellt. Das Ende der Datensätze ist durch ein EOF-Zeichen gekennzeichnet (Abb. 187). Die Angaben im Haupt-, Zylinder- oder Spurindex sind z. B. durch ein FFF-Zeichen abgeschlossen.
Nachdem man sich von der Wirksamkeit des Adreß-Index-Systems bei der logisch-fortlaufenden Speicherungsform überzeugt hat, kann man ihm eine gewisse Anerkennung nicht versagen: „Raffiniert ausgedacht". Doch für den Programmierer erhebt sich nun die Frage, welche Mühe er sich machen muß, um einen solchen Datenbestand samt Adreß-Index anzulegen. Um es gleich vorwegzunehmen, der Aufwand ist unerwartet gering; eine andere Antwort hätte sicherlich auch mehr als abschreckend gewirkt.
Der Aufbau des Adreß-Indexes vollzieht sich völlig automatisch ohne Zutun des Programmierers. In der Dateibeschreibung sind lediglich Angaben über die Lage (Grenzen) des Haupt- und Zylinderindex zu machen. Außerdem ist anzugeben, in welchem Bereich die Datensätze gespeichert werden sollen (auf den Spurindex braucht in diesem Zusammenhang nicht hingewiesen zu werden; er steht später immer in der Spur 0 eines Datenzylinders). Desweiteren ist in der Dateibeschreibung auch der symbolische Name des Schlüsselfeldes im Hauptspeicher zu vereinbaren, seine Größe wird unter Definitionen festgelegt. Alle weiteren Angaben gehen aus der folgenden Aufstellung hervor:

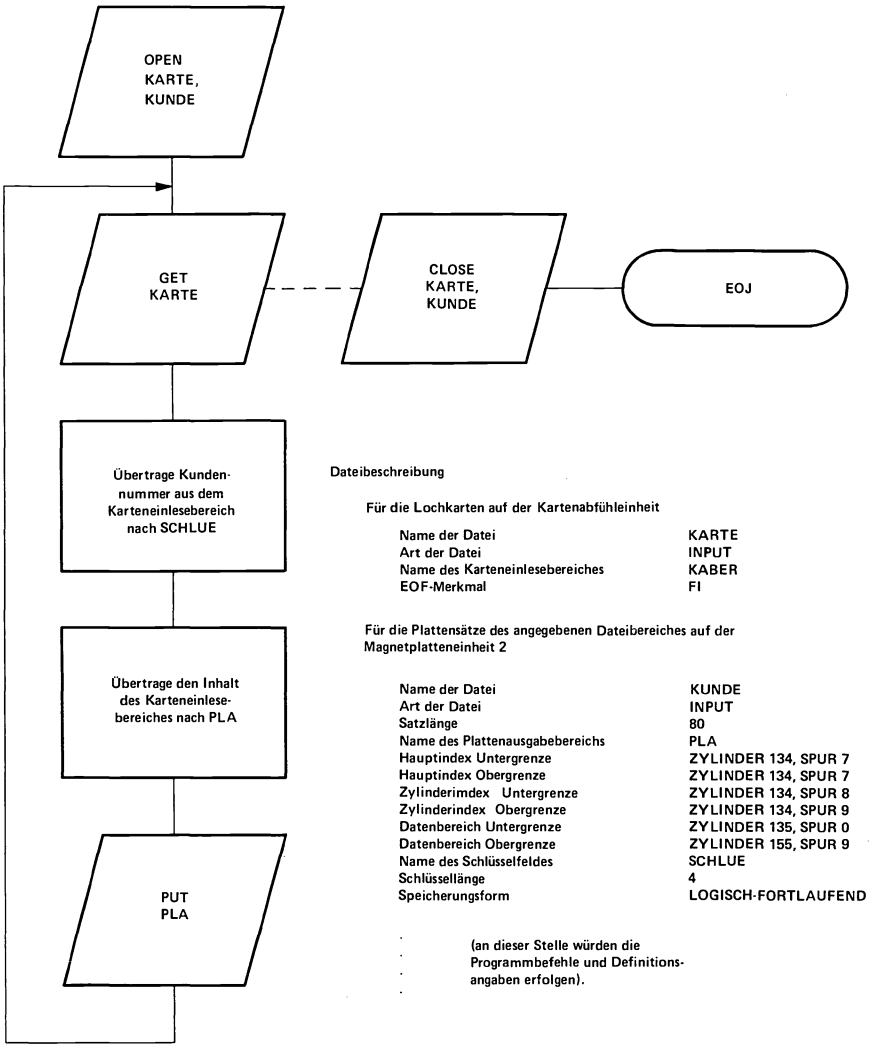
Eine Dateibeschreibung für einen logisch-fortlaufend gespeicherten Datenbestand mit Adreß-Index umfaßt folgende Angaben:

Name der Datei
Art der Datei
Satzlänge
Name des Plattenausgabebereiches
Hauptindex Untergrenze
Hauptindex Obergrenze

* Das Lesen von Satzadressen und Indexangaben erfolgt automatisch. Die erfaßten Werte werden in speziellen elektronischen Schaltungen ausgewertet.
Verwechseln Sie dieses Lesen bitte nicht mit dem Einlesen eines Datensatzes in den Einlesebereich des Hauptspeichers.

Zylinderindex Untergrenze
Zylinderindex Obergrenze
Datenbereich Untergrenze
Datenbereich Obergrenze
Name des Schlüsselfeldes
Schlüssellänge
Speicherungsform

In Abb. 191 ist das Programm zum Anlegen einer Kundendatei (Abb. 187) gezeigt. Als unbedingte Voraussetzung ist zu beachten, daß die einzugebenden Lochkarten



Programm zum Anlegen einer logisch-fortlaufend gespeicherten Kunden-Datei
Abb. 191

(Abb. 186) nach aufsteigender Kundennummer sortiert sein müssen. Es mutet fast unwahrscheinlich an, daß die Angaben in der Dateibeschreibung und die wenigen Programmbefehle ausreichen, um beim Schreiben (Anlegen) der einzelnen Datensätze zugleich ein derart kompliziertes Adreß-Index-System zu erzeugen, wie es in Abb. 187 dargestellt ist.* Im Anschluß an den letzten Datensatz wird automatisch ein EOF-Datei-Ende-Merkmal gesetzt.

* Das Adreß-Index-System ist in Wirklichkeit komplizierter als dargestellt aufgebaut. Seine prinzipielle Wirksamkeit bleibt davon unberührt.

B. Verarbeitungsformen von Magnetplatten-Dateien

I. Die starr-fortlaufende Verarbeitungsform
II. Die wahlweise Verarbeitungsform
III. Die logisch-fortlaufende Verarbeitungsform

Abgesehen von Ein/Ausgabeinstruktionen bleibt der Wirkungsbereich von Programmbefehlen (Übertrage, Vergleiche, Addiere etc.) allein auf den Hauptspeicher beschränkt. Die eigentliche Verarbeitung von Daten ist demnach nur möglich, wenn sie zuvor in den Hauptspeicher eingelesen wurden. Dort unterliegen sie der Einwirkung des Programmes – dort können sie in andere Felder übertragen, durch Rechnungsoperationen verknpüft, kurz gesagt, verarbeitet werden.
Der Begriff „Verarbeitungsformen" hat zu dieser Tatsache eine nur sehr weitläufige Beziehung. Er bezieht sich auf die verschiedenen Möglichkeiten der Aufeinanderfolge einzulesender Datensätze. Bei Lochkarten und dem Magnetband ergibt sie sich von selbst. Lochkarten können nur in der gleichen Reihenfolge gelesen werden, wie sie dem Karten-Leser zugeführt wurden und für Band-Dateien gilt ebenfalls das Prinzip einer seriellen, satzweisen Verarbeitung.
Das Einlesen von Plattensätzen unterliegt nicht einem solchen starren Schema. Der Lesebefehl kann für jede beliebige Satzadresse wirksam werden, unabhängig von der Lage des zuvor gelesenen Datensatzes. Durch diese Flexibilität ist man nicht wie beim Magnetband an eine unmittelbar anschließende Satzfolge gebunden. Jetzt ist es vielmehr denkbar, daß bspws. nach dem 83. Satz (nicht etwa der 84., sondern) je nach Anforderung der 7. oder auch 127. Satz eingelesen werden kann.
Eine solche Möglichkeit des wahlfreien Zugriffs erweist sich in vielen Fällen als sehr vorteilhaft; mitunter ist jedoch auch ein „stetiges Nacheinander" der eingelesenen Datensätze erwünscht (Satz Nr. 0, Satz Nr. 1, Satz Nr. 2 usw.).
Zur Kennzeichnung dieser Möglichkeiten wurden die Begriffe

einer starr-fortlaufenden,
einer wahlfreien und
einer logisch-fortlaufenden Verarbeitungsform

eingeführt.
Je nach Aufgabe des Programmes greift man auf eine dieser Verarbeitungsformen zurück. Dabei kann es vorkommen, daß die gleiche Platten-Datei einmal starr-fortlaufend, und ein anderes Mal wahlfrei verarbeitet werden soll. Eine derartige Forderung läßt sich aber nur dann erfüllen, wenn für den Datenbestand auch eine zweckentsprechende Speicherungsform gewählt wurde. Es ist nämlich durchaus nicht selbstverständlich, daß jede Speicherungsform auch jede nur denkbare Verarbeitungsform zuläßt.

1. Verarbeitungsformen für einen starr-fortlaufend gespeicherten Datenbestand

a) Die starr-fortlaufende Verarbeitungsform für einen starr-fortlaufend gespeicherten Datenbestand

Dieser Begriff deutet an, daß während eines Programmlaufes alle Datensätze in einer Reihenfolge eingelesen werden, die ihrer Anordnung auf der Platte entspricht.

Ein fortlaufend gespeicherter Datenbestand kann nur als ganzes betrachtet (ab Untergrenze satzweise bis zu seinem Ende) verarbeitet werden.
Der Programmieraufwand ist in einem solchen Falle gering. In der Datei-Beschreibung ist u. a. aufzuführen, welche Speicherungsform der zu verarbeitende Datenbestand hat (starr-fortlaufend) und welcher Verarbeitungsform (in diesem Falle ebenfalls starr-fortlaufend) er unterliegen soll. Im Programm braucht für das Einlesen eines Datensatzes lediglich ein Lesebefehl GET (Dateiname) gegeben zu werden; die explizite Angabe einer Plattenadresse entfällt.
Diese Vereinfachung ist dadurch begründet, daß die Ein/Ausgabe-Steuerung die Einstellung des S-L-Kopfes von Satz zu Satz selbst übernimmt. Wird der Lesebefehl das allererste Mal erteilt, so bezieht er sich auf den Datensatz an der unteren Dateigrenze; beim zweiten Mal erfolgt das Einlesen des unmittelbar benachbarten Satzes mit der nächsthöheren Satznummer usw. usw.. Nach jedem Lesebefehl erhöht die Ein/Ausgabe-Steuerung ohne unser Zutun die Plattenadresse um 1 und bewirkt die entsprechende Einstellung des S-L-Kopfes. Die Datensätze werden demnach in starr-fortlaufender Reihenfolge gelesen. Dieser Vorgang ist erst dann beendet, wenn auf der Platte das EOF-Datei-Ende-Kennzeichnen festgestellt wird. In diesem Falle verzweigt das Programm automatisch zu dem in der Datei-Beschreibung angegebenen EOF-Merkmal.
In Abb. 192 ist das Schema eines Programmes für das Auflisten aller Vertretersätze (Abb. 178, Speicherungsform: starr-fortlaufend) mit einer möglichen zusätzlichen Zwölfer-Zonung (entsprechend einer 12-er Lochung in einer Lochkarte) in Stelle 4 gezeigt. (Eine Zwölfer-Zonung in Stelle 4 soll besonders erfolgreicher Vertreter andeuten). Diese Aufgabe setzt eine starr-fortlaufende Verarbeitung voraus. Nach dem Einlesen eines Satzes folgt die Abfrage auf eine 12-Zone in der 4. Stelle des Einlesebereiches. Bei Bejahung ist dieser Satz aufzulisten, andernfalls kehrt man zum Lesebefehl zurück. Das Einlesen der Datensätze ist beendet, wenn das EOF-Datei-Ende-Kennzeichen erreicht ist. In diesem Falle verzweigt das Programm automatisch zu der in der Dateibeschreibung angegebenen EOF-Adresse FINITO – nach dem „Abschließen" aller beteiligten Dateien ist das Programm beendet.
Beim Anlegen der Vertreter-Datei hatte man die Wahl einer sortierten oder unsortierten Lochkarteneingabe. Die erstere Möglichkeit würde jetzt den Vorteil bieten, daß die betreffenden Vertretersätze in der Reihenfolge aufsteigender Vertreternummer auf der Liste erscheinen.
Es erweist sich also als sehr vorteilhaft, wenn ein starr-fortlaufend gespeicherter Datenbestand nach dem maßgebenden Ordnungsbegriff sortiert ist.

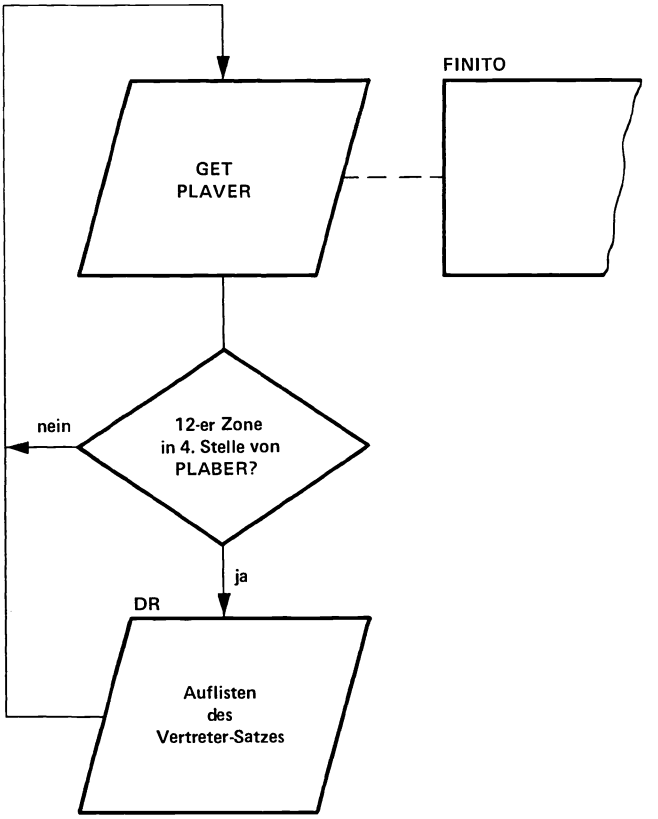
Datenbeschreibung

Für die Plattensätze des angegebenen Dateibereiches auf der Magnetplatten-Einheit 2

Name der Datei	PLAVER
Art der Datei	INPUT
Satzlänge	80
Name des Platteneinlesebereiches	PLABER
Untergrenze	ZYLINDER 128, SPUR 0
Obergrenze	ZYLINDER 131, SPUR 9
EOF-Merkmal	FINITO

Speicherungsform	STARR-FORTLAUFEND
Verarbeitungsform	STARR-FORTLAUFEND

. (an dieser Stelle müßten die Beschreibung
. der Drucker Datei, Programmbefehle und
. Defintionsangaben folgen)
.



Auszugsweise Darstellung eines Programmablaufplanes, nach dem ein starr-fortlaufend gespeicherter Datenbestand einer starr-fortlaufenden Verarbeitung unterzogen wird.

Abb. 192

b) Der Begriff des Updatens

Updaten (gesprochen: „Λpde:ten") hat sich als fremdsprachliche Bezeichnung im Wortschatz der elektronischen Datenverarbeitung einen festen Platz erobert. Eine gewisse

Verwandtschaft mit üblichen Floskel „up to date" (zeitgemäß, auf der Höhe) ist unverkennbar. Und so bedeutet updaten in diesem Sinne auch nichts anderes als „auf den neuesten Stand bringen". In der Verarbeitung von Plattendateien ist damit die Tatsache verbunden, einen bestimmten Datensatz einzulesen, im Hauptspeicher zu verändern (auf den neuesten Stand zu bringen) und auf seine ursprüngliche Plattenposition zurückzuschreiben. Beim Magnetband war so etwas undenkbar: Nach Ablauf eines Lesebefehles hätte sich das Band bereits um die Länge des soeben eingelesenen Satzes am S-L-Kopfe vorbeibewegt; die ursprüngliche Position des Bandsatzes ist somit für einen nachfolgenden Schreibbefehl nicht mehr erreichbar, es sei denn, das Band würde zuvor um Satzlänge zurückgespult werden, was im Normalfall nicht möglich ist.
Bei der Magnetplatte hingegen liegen die Verhältnisse günstiger. Unter dem S-L-Kopf rotiert ständig die ausgewählte Datenspur. Dabei wird im geeigneten Moment der gewünschte Satz gelesen. Sobald er im Hauptspeicher per Programmbefehl „auf den neuesten Stand" gebracht wurde, kann er wieder auf seinen ursprünglichen Platz auf der Platte zurückgeschrieben werden. Der S-L-Kopf muß dabei auf die gleiche Satzadresse (wie ursprünglich beim Einlesen) ausgerichtet sein. Für diese Forderung ist automatisch gesorgt, wenn in der Dateibeschreibung der Vermerk „Updaten" erscheint. Beim Updaten wird der gleiche Hauptspeicherbereich für das Einlesen und die Ausgabe eines Satzes benutzt.
In Abb. 193 ist ein Programmschema gezeigt, nach dem die Vertretersätze eines starr-fortlaufend gespeicherten Datenbestandes einer starr-fortlaufenden Verarbeitung mit Updaten unterzogen werden. Danach sollen alle diejenigen Vertretersätze mit einem Umsatz (Stelle 65—72) größer als 15 000 DM eine 12-er Zone in Stelle 4 erhalten.
Nach dem Einlesen eines Vertretersatzes wird gefragt, ob der Umsatz größer als 15 000 ist. Bei einer bejahenden Antwort wird in die 4. Stelle des Plattenein/ausgabebereiches PLUP eine 12-er Zone übertragen. Damit ist jedoch der Vertretersatz bereits auf den neuesten Stand gebracht, er muß nur noch auf die Platte zurückgeschrieben werden. Ein kleinerer Umsatz als 15 000 DM in einem Vertretersatz macht sein Updaten unnötig. Es kann sofort zum Einleiten des nächsten Vertretersatzes übergegangen werden.

2. Verarbeitungsformen eines gestreut gespeicherten Datenbestandes

a) Die wahlweise Verarbeitungsform eines gestreut gespeicherten Datenbestandes

Diese Verarbeitungsform läßt sich besonders gut am Beispiel der Artikel-Datei (Abb. 181) erläutern. Es ist bereits bekannt, daß jeder Artikel-Satz an einer ganz bestimmten Stelle des Dateibereiches gespeichert ist. Seine Plattenadresse läßt sich aus der Artikelnummer errechnen. Diesem Umstand allein ist es zu verdanken, daß während der wahlweisen Verarbeitung der Artikel-Datei mal der eine, mal der andere Datensatz eingelesen werden kann. Von einer bestimmten Reihenfolge kann dabei keine Rede mehr sein.

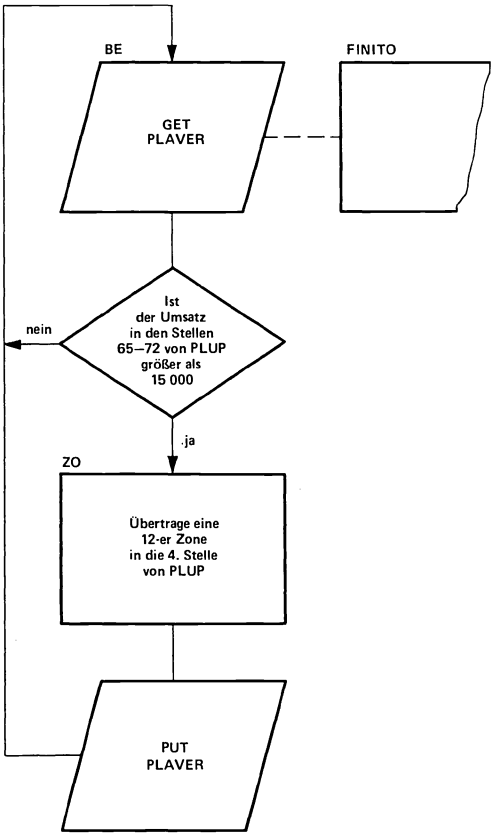
Dateibeschreibung

Für die Plattensätze des angegebenen Dateibereiches auf der Magnetplatten-Einheit 2

Name der Datei	PLAVER
Art der Datei	INPUT, OUTPUT

Satzlänge	80
Name d. Plattenein/ausgabebereiches	PLUP
Untergrenze	ZYLINDER 128, SPUR 0
Obergrenze	ZYLINDER 131, SPUR 9
EOF-Merkmal	FINITO
Speicherungsform	STARR-FORTLAUFEND
Verarbeitungsform	STARR-FORTLAUFEND MIT UPDATEN

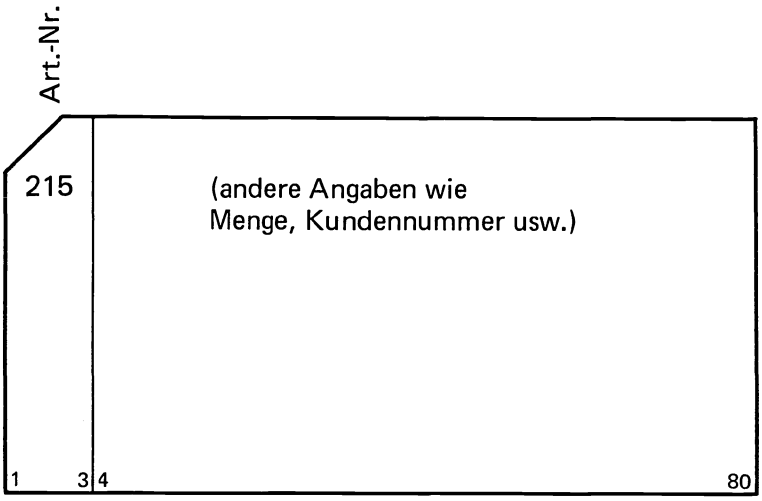
. (an dieser Stelle müßten die
. Programmbefehle und Defi-
. nitionsangaben folgen)
.



Auszugsweise Darstellung eines Programmablaufplanes, nach dem ein starr-fortlaufend gespeicherter Datenbestand einer starr-fort laufenden Verarbeitung mit UPDATEN unterzogen wird.

Abb. 193

Die Angabe, welcher Artikelsatz jeweils einzulesen ist, stammt z. B. aus einer Bestellkarte (Abb. 194). Dort ist u. a. nur die Artikelnummer abgelocht; diese Angabe allein ist jedoch für eine Rechnungsschreibung unzureichend. Die zusätzlich erforderlichen Hinweise, wie Artikelbezeichnung, Preise etc. sind nicht in der Bestellkarte, wohl aber im entsprechenden Artikelsatz enthalten. Um auf sie zurückgreifen zu können, muß dieser Artikelsatz in den Hauptspeicher eingelesen werden.
Nach dem Lesen einer Bestell-Lochkarte wird die anfallende Artikelnummer in die Plattenadresse des zugehörigen Artikelsatzes umgerechnet (Umrechnungsroutine Abb. 184). Diese Adresse wird dann in das Plattenadreßfeld übertragen – der S-L-Kopf stellt sich daraufhin auf die angegebene Position ein, so daß der gewünschte Artikelsatz durch den GET-Befehl in den Hautspeicher gelangt.



Bestell-Lochkarte

Abb. 194

Für eine solche wahlweise Verarbeitung eines gestreuten Datenbestandes sind verschiedene Eintragungen in der Dateibeschreibung vorzunehmen. Sie verstehen sich immer im Zusammenhang mit dem Programmablaufplan, wie in Abb. 195 gezeigt wird.

b) Die starr-fortlaufende Verarbeitungsform eines gestreut gespeicherten Datenbestandes

Von dieser Möglichkeit wird z. B. dann Gebrauch gemacht, wenn alle Artikelsätze (Abb. 181) nacheinander aufzulisten sind. Zu diesem Zweck müssen die einzelnen Datensätze in ihrer physischen Reihenfolge nacheinander eingelesen werden. Dabei kann es allerdings vorkommen, daß wegen der vorhandenen Satzlücken (nicht jeder Platz ist mit einem Artikelsatz belegt) auch „Leersätze" in den Hauptspeicher gelangen. Es wäre natürlich unsinnig, Leersätze aufzulisten (dabei ergäbe sich jeweils eine Leerzeile). Aus diesem Grunde ist im Programm (Abb. 196) die Frage nach einem Blank in der ersten Stelle des Einlesebereiches vorgesehen. Eine bejahende Antwort deutet an, daß der

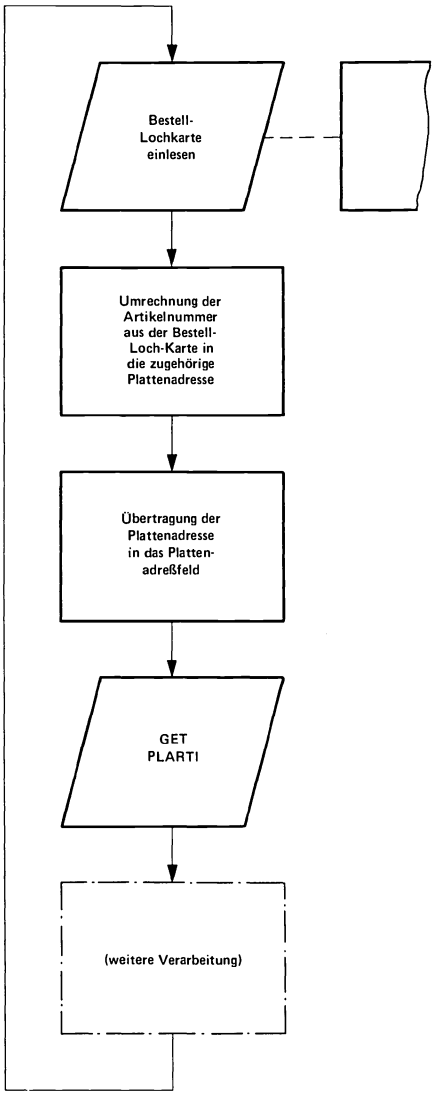
soeben eingelesene „Satz" ein „Leersatz" ist – folglich kann sofort zum Einlesen des nächsten Plattensatzes übergegangen werden.
Die prinzipiellen Angaben, die im Falle einer starr-fortlaufenden Verarbeitung eines gestreut gespeicherten Datenbestandes in der Dateibeschreibung und im Programmablaufplan aufzunehmen sind, werden am Beispiel des Programmes „Artikelsätze listen" in Abb. 196 angeführt.

Dateibeschreibung

Für die Plattensätze des angegebenen Dateibereiches auf der Magnetplatten-Einheit 2

Name der Datei	PLARTI
Art der Datei	INPUT
Satzlänge	80
Name des Platteneinlesebereiches	PLAB
Untergrenze	ZYLINDER 132, SPUR 0
Obergrenze	ZYLINDER 133, SPUR 9
Name des Plattenadreßfeldes	PLAADR
Speicherungsform	GESTREUT
Verarbeitungsform	WAHLWEISE

. (an dieser Stelle müßten die Beschreibung
. weiterer Dateien, Programmbefehle und
. Defintionsangaben folgen)
.



Auszugsweise Darstellung eines Programmablaufplanes, nach dem ein gestreuter Datenbestand wahlweise verarbeitet wird.

Abb. 195

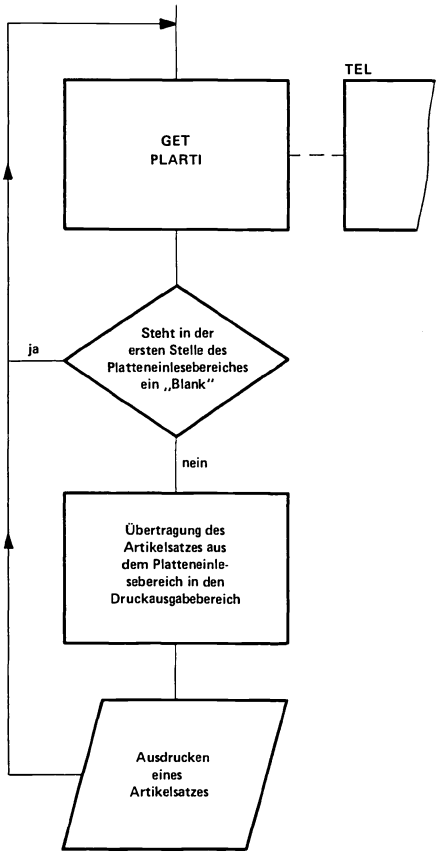
Dateibeschreibung

Für die Plattensätze des angegebenen Dateibereiches auf der Magnetplatten-Einheit 2

Name der Datei	PLARTI
Art der Datei	INPUT

Satzlänge	80
Name des Platteneinlesebereiches	PLAB
Untergrenze	ZYLINDER 132, SPUR 0
Obergrenze	ZYLINDER 133, SPUR 9
EOF-Merkmal	TEL
Speicherungsform	GESTREUT
Verarbeitungsform	STARR-FORTLAUFEND

. (an dieser Stelle müßten die Beschreibungen für weitere Dateien, Programmbefehle und Definitions-
. angaben folgen).



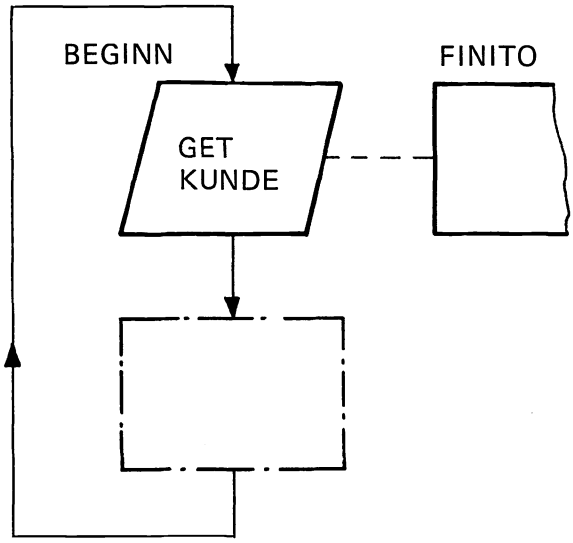
Auszugsweise Darstellung eines Programmablaufplanes, nach dem ein gestreuter Datenbestand starr-fortlaufend verarbeitet wird.

Abb. 196

3. Verarbeitungsformen eines logisch-fortlaufend gespeicherten Datenbestandes

a) Die starr-fortlaufende Verarbeitungsform eines logisch-fortlaufend gespeicherten Datenbestandes

Die starr-fortlaufende Verarbeitungsform bezieht sich auf die Gesamtheit aller Datensätze. Diese werden in einer Reihenfolge eingelesen, die ihrer (physischen) Anordnung auf der Magnetplatte entspricht. Bei Erreichung des „EOF"-Zeichens (Kennzeichnung des Datei-Endes) verzweigt das Programm automatisch zu der in der Dateibeschreibung angegebenen EOF-Adresse.
In Abb. 197 ist der Ausschnitt eines Programmablaufplanes gezeigt, nach dem eine starr-fortlaufende Verarbeitung der logisch-fortlaufend gespeicherten Kunden-Datei (Abb. 187) erfolgt. Der strich-punktiert ausgeführte Block symbolisiert die eigentliche Verarbeitung eines eingelesenen Datensatzes, die in diesem Zusammenhang uninteressant ist.
Sobald die Kunden-Datei Zugänge erfahren hat (Abb. 200), entspricht die physische nicht mehr der logischen Satzfolge. Dieser Umstand wirkt sich bei der starr-fortlaufenden Verarbeitung dahingehend aus, daß die Kundensätze nicht in der Reihenfolge aufsteigender Ordnungsnummern (Kundennummer) eingelesen werden.



Auszugsweise Darstellung eines Programmablaufplanes, nach dem ein logisch-fortlaufend gespeicherter Datenbestand starr-fortlaufend verarbeitet wird.

Abb. 197

Dateibeschreibung

Für die Plattensätze des angegebenen Dateibereiches auf der Magnetplatteneinheit 2

Name der Datei	KUNDE
Art der Datei	INPUT
Satzlänge	80
Name des Platteneinlesebereiches	PLA
Datenbereich Untergrenze	ZYLINDER 135, SPUR 0
Datenbereich Obergrenze	ZYLINDER 155, SPUR 9
EOF-Merkmal	FINITO
Speicherungsform	LOGISCH-FORTLAUFEND
Verarbeitungsform	STARR-FORTLAUFEND

b) Wahlweise Verarbeitung eines logisch-fortlaufend gespeicherten Datenbestandes

Ihr gebührt eine besondere Beachtung; sie ermöglicht das Einlesen eines bestimmten Datensatzes. Dieser wird nicht etwa durch Nennung seiner Plattenadresse, sondern durch Angabe seines Ordnungsbegriffes, z. B. der Kundennummer, gekennzeichnet.
Im Rahmen einer Rechnungsschreibung müßte für die in der Bestellkarte (Abb. 194) angegebene Kundennummer der zugehörige Kundensatz zur Verfügung gestellt (eingelesen) werden. Dadurch erhält man die für den Rechnungskopf benötigte Information, wie die Anschrift etc.
Der Programmierer hat zur Erfüllung einer solchen Aufgabe lediglich den Ordnungsbegriff des gewünschten Satzes in das Schlüsselfeld zu übertragen. Der daran anschließende Lesebefehl besorgt das Einlesen des betreffenden Datensatzes in den Einlesebereich des Kernspeichers.
Der symbolische Name des Schlüsselfeldes wird in der Dateibeschreibung festgelegt (Abb. 198), seine Größe unter Defintionen vereinbart.
In Abb. 198 sind als Beispiel für einen wahlweise zu verarbeitenden logisch-fortlaufend gespeicherten Datenbestand die Dateibeschreibung für die Kunden-Datei (nach Abb. 187) und der für die Einleseroutine wichtige Teil des Blockdiagrammes angegeben. Das Einlesen des gewünschten Satzes erfolgt automatisch. Er wird vom S-L-Kopf unter Benutzung des Adreß-Indexes aufgesucht. Das Programm kehrt anschließend zum Einlesen einer neuen Bestell-Lochkarte zurück. Auf Grund der sie enthaltenden Kundennummer wird wieder der zugehörige Kundensatz in den Hauptspeicher eingelesen usw. usw.. Der strich-punktierte Block des Programmablaufplanes deutet die weitere Verarbeitung der Daten des soeben eingelesenen Satzes an.

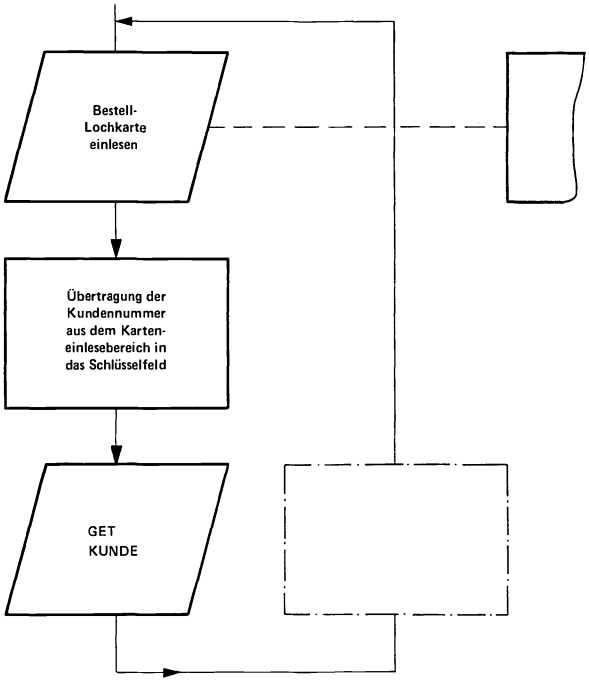
Dateibeschreibung

Für die Plattensätze des angegebenen Dateibereiches auf der Magnetplatteneinheit 2

Name der Datei	KUNDE
Art der Datei	INPUT
Satzlänge	80
Name des Platteneinlesebereiches	PLA

Hauptindex Untergrenze	ZYLINDER 134, SPUR 7
Hauptindex Obergrenze	ZYLINDER 134, SPUR 7
Zylinderindex Untergrenze	ZYLINDER 134, SPUR 8
Zylinderindex Obergrenze	ZYLINDER 134, SPUR 9
Datenbereich Untergrenze	ZYLINDER 135, SPUR 0
Datenbereich Obergrenze	ZYLINDER 155, SPUR 9
Name des Schlüsselfeldes	SCHLUE
Schlüssellänge	4
Speicherungsform	LOGISCH-FORTLAUFEND
Verarbeitungsform	WAHLWEISE

. (an dieser Stelle würden die Beschreibungen
. weiterer Dateien, Programmbefehle und
. Definitionsangaben erfolgen)
.



Auszugsweise Darstellung eines Programmablaufplanes, nach dem ein logisch-fortlaufend gespeicherter Datenbestand wahlweise verarbeitet wird.

Abb. 198

c) Die logisch-fortlaufende Verarbeitungsform eines logisch-fortlaufend gespeicherten Datenbestandes

Diese Verarbeitungsform wird immer mehr dann gewählt, wenn sämtliche Sätze eines Datenbestandes in einer ganz bestimmten Reihenfolge nacheinander gelesen werden sollen.
Dieser Reihenfolge liegt diesmal allerdings nicht die physische Satzordnung zugrunde, wie sie bei der starr-fortlaufenden Verarbeitungsform ausschlaggebend war.
Es ist nämlich durchaus auch möglich, einen völlig anderen Gesichtspunkt zu wählen und die Datensätze, unbeachtet ihres Platzes auf der Platte, in der Reihenfolge ihres aufsteigenden Ordnungsbegriffes (bspws. der Kundennummer) einzulesen.
Von dieser Möglichkeit, der logisch-fortlaufenden Verarbeitungsform, kann nur dann Gebrauch gemacht werden, wenn die vorliegende Datei logisch-fortlaufend gespeichert ist.
Es ist zunächst mit Recht einzuwenden, daß — gemessen an den bisherigen Ausführungen — eine unterschiedliche Auswirkung zwischen einer starr- und logisch-fortlaufenden Verarbeitungsform noch nicht erkennbar ist.
Denn immerhin war beim Anlegen einer logisch-fortlaufend gespeicherten Datei zu beachten, daß die Sätze unbedingt nach aufsteigendem Ordnungsbegriff auf die Platte gelangen. Ihre physische Anordnung entspricht somit auch der Reihenfolge nach aufsteigendem Ordnungsbegriff.
Ihre Einlesefolge wäre demnach bei der starr- oder logisch-fortlaufenden Verarbeitungsform die gleiche. Die Existenzberechtigung der logisch-fortlaufenden Verarbeitungsform scheint somit noch nicht ausreichend begründet zu sein.
Sie erlangt auch erst in einem anderen Zusammenhang eine Bedeutung, nämlich dann, wenn der logisch-fortlaufend gespeicherte Datenbestand Zugänge aufgenommen hat. Dieser Punkt verdient eine ausführlichere Betrachtung — er ist von grundlegender Wichtigkeit.
Über Zugänge, das Einfügen neu aufzunehmender Sätze in einen bereits bestehenden Datenbestand, ist in der Übungsfolge 14 schon etwas gesagt worden. Die Aufnahme von Zugängen war nur dann möglich, wenn die gesamte, bereits bestehende Datei gelesen und unter Berücksichtigung der Zugänge neu geschrieben wurde. Diese Ausführungen beschränkten sich jedoch nur auf Magnetband-Dateien.
Bei Platte-Dateien wird die Aufnahme von Zugängen auf recht unterschiedliche Art und Weise gelöst; sie richtet sich immer nach der vorliegenden Speicherungsform. Zugänge werden immer durch ein entsprechendes Programm in eine bereits bestehende Datei aufgenommen.

Erfassung von Zugängen

Erfassung von Zugängen bei einem starr-fortlaufend gespeichertem Datenbestand

Hier läßt sich eine gewisse Analogie zu Magnetband-Dateien nicht verkennen. Falls der starr-fortlaufend gespeicherte Datenbestand nach aufsteigendem Ordnungsbegriff (bspws. der Vertreternummer) angelegt wurde, so wird es auch erwünscht sein, die Zugänge an den passenden Stellen einzuordnen — z. B. den neu aufzunehmenden Vertretersatz mit der Vertreternummer 254 zwischen die bereits bestehenden Vertretersätze mit den Vertreternummern 233 und 280 „einzuschieben". Das ist jedoch nur dann möglich, wenn die gesamte, bereits bestehende Vertreter-Datei (starr-fortlaufend) ge-

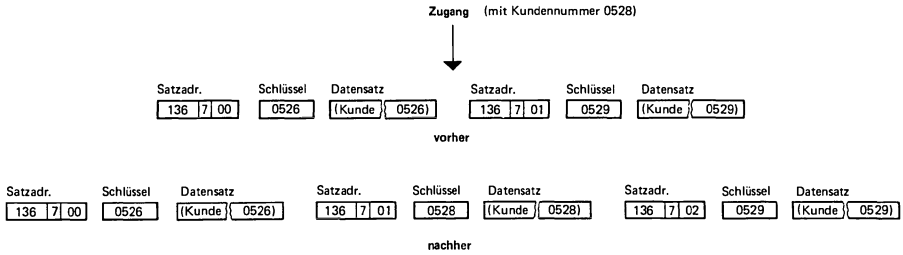
lesen und unter Berücksichtigung der Zugänge neu geschrieben wird. Für die „Vertreter-Datei neu" ist ein entsprechender Plattenbereich vorzusehen.

Erfassung von Zugängen bei einem gestreut gespeicherten Datenbestand

Wie bereits erwähnt, besteht bei der gestreuten Speicherungsform zwischen dem Ordnungsbegriff eines Datensatzes, bspws. der Artikelnummer und seiner Plattenadresse, ein Zusammenhang. Die Plattenadresse läßt sich mit Hilfe einer Umrechnungsroutine (Abb. 184) im Programm ermitteln.
Innerhalb der gesamten Artikel-Datei ist von vornherein für jede theoretisch denkbare Artikelnummer (001—500) ein bestimmter Platz auf der Platte reserviert. Jedoch wurden nur alle die Plätze belegt, für die es eine im Geschäftsbetrieb verwendete Artikelnummer gab.
Bei einer nicht hundertprozentigen Belegungsdichte bleiben bestimmte Plätze auf der Platte frei (Satzlücken).
Es ist denkbar, daß zu einem späteren Zeitpunkt Neuzugänge bisher nicht benutzte Artikelnummern erhalten. Die dazugehörigen Artikelsätze müssen dann in die schon bestehende Datei (Abb. 181) aufgenommen werden. Dies erfordert keinen zusätzlichen Programmier-Aufwand. Das bereits existierende Programm „ARTIKELSÄTZE-ANLEGEN" (Abb. 184) kann auch dem Einfügen der Zugänge dienen: jeder neue Artikelsatz gelangt auf Grund der Umrechnungsroutine an seinen vorgeschriebenen Platz auf der Platte. Ein Umschreiben der Datei ist somit nicht erforderlich.

Erfassung von Zugängen bei der logisch-fortlaufenden Speicherungsform

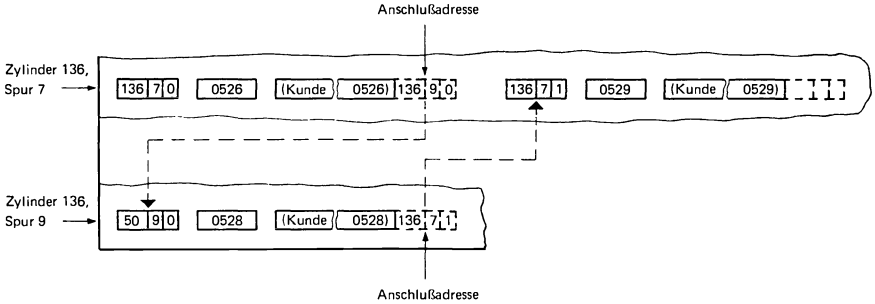
Aus Abb. 187 geht hervor, daß bei einem logisch-fortlaufend gespeicherten Datenbestand alle Sätze unmittelbar aufeinanderfolgen. Ihre Anordnung auf der Platte entspricht der Sortierfolge des Ordnungsbegriffes der Kundennummer. Die Kundennummer ist in dem sogenannten Schlüssel dem eigentlichen Datensatz vorangestellt (Abb. 188). Der Schlüssel eines vorangehenden Satzes ist immer niedriger als der des nachfolgenden. Im Laufe der Zeit werden jedoch neue Kunden gewonnen, für die ebenfalls Kundensätze anzulegen sind. Diese sollen in die bereits bestehende Kunden-Datei „eingeordnet" werden. Es liegt die Vermutung nahe, daß zu diesem Zweck die gesamte Datei umgeschrieben werden müßte. Denn nur so wäre es möglich, z. B. den Neuzugang mit der Kundennummer 0528 zwischen die schon bestehenden Sätze mit den Kundennummern 0526 und 0529 physisch richtig „einzuschieben" (Abb. 199).



Physische Einordnung eines Zuganges

Abb. 199

Die Konzeption der logisch-fortlaufenden Speicherungsform mit Adreßindex sieht jedoch eine besondere Art der Erfassung von Zugängen vor, die ein Umschreiben der Datei nicht erforderlich macht.
Danach werden die neu aufzunehmenden Datensätze nicht physisch, sondern logisch eingeordnet. Der Programmierer gibt für einen neu aufzunehmenden Datensatz lediglich einen Schreibbefehl; alles andere erfolgt automatisch unter der Ein/Ausgabe-Steuerung.
Vereinfachtes Beispiel:
Zur Aufnahme der Zugänge* bleibt in jedem Zylinder die Spur 9 reserviert. Dorthin werden die neuen Datensätze geschrieben. Ihre logische Einordnung in den bereits bestehenden Datenbestand geschieht durch zusätzliche Eintragungen im Adreßindex (hier nicht erörtert) und durch sogenannte Anschlußadressen. Diese Eintragungen dienen einem besonderen Zweck. Zur Erläuterung sei angenommen (Abb. 200), daß zwischen die bereits bestehenden Datensätze mit den Kundennummern 0526 und 0529 ein neuer Kundensatz mit der Kundennummer 0528 logisch richtig eingeordnet worden wäre. Letzterer steht in der Spur 9.
Bei der späteren Verarbeitung eines logisch-fortlaufenden Datenbestandes inclusive seiner Zugänge wird mitunter darauf Wert gelegt, daß das Einlesen aller Datensätze in der Reihenfolge nach aufsteigenden Ordnungsnummern (Kundennummern) erfolgt. (Man spricht in diesem Zusammenhang von einer logisch-fortlaufenden Verarbeitung). Dieser Forderung kann mit Hilfe der Anschlußadressen entsprochen werden.



Logische Einordnung eines Zuganges mit Hilfe von Anschlußadressen

Abb. 200

Nach dem Lesen des Datensatzes mit der Kundennummer 0526 stellt sich der S-L-Kopf automatisch auf die genannte Anschlußadresse „Zylinder 136 Spur 9 Satz 0" ein. Dort steht der Kundensatz mit dem nächsthöheren Ordnungsbegriff. Er wird eingelesen. Seine Anschlußadresse „Zylinder 136 Spur 7 Satz 1" verweist wiederum auf die Stelle der Magnetplatte, an der der nächste einzulesende Satz steht. Durch dieses geschickte „Hin und Her" werden die Datensätze in der Reihenfolge nach aufsteigendem Ordnungsbegriff, sprich in der logisch richtigen Reihenfolge, gelesen.
Das in Abb. 201 gezeigte Programm ermöglicht zum Beispiel das Auflisten von Kundensätzen nach aufsteigender Kundennummer selbst dann noch, wenn die Kunden-Datei (Abb. 187, 190) in den dafür vorgesehenen 9-er Spuren Zugänge erfahren hat.

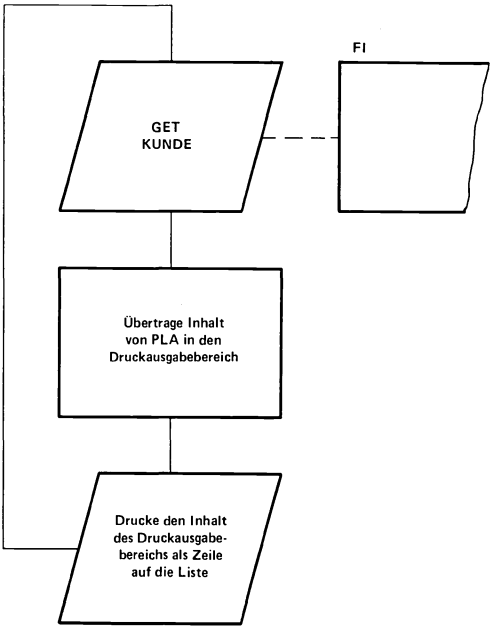
* Es ist strikt darauf zu achten, daß die Lochkarten für die neu aufzunehmenden Zugänge in sortierter Reihenfolge dem Karten-Leser zuzuführen sind.

Dateibeschreibung

Für die Plattensätze des angegebenen Dateibereiches auf der Magnetplatteneinheit 2

Name der Datei	KUNDE
Art der Datei	INPUT
Satzlänge	80
Name des Platteneinlesebereiches	PLA
Hauptindex Untergrenze	ZYLINDER 134, SPUR 7
Hauptindex Obergrenze	ZYLINDER 134, SPUR 7
Zylinderindex Untergrenze	ZYLINDER 134, SPUR 8
Zylinderindex Obergrenze	ZYLINDER 134, SPUR 9
Datenbereich Untergrenze	ZYLINDER 135, SPUR 0
Datenbereich Obergrenze	ZYLINDER 155, SPUR 9
Speicherungsform	LOGISCH-FORTLAUFEND
Verarbeitungsform	LOGISCH-FORTLAUFEND
EOF-Merkmal	FI

. (an dieser Stelle müßten die
. Beschreibung der Drucker Datei,
. Programmbefehle und Definitions-
. angaben folgen)



Auszugsweise Darstellung eines Programmablaufplanes, nach dem ein logisch-fortlaufend gespeicherter Datenbestand logisch-fortlaufend verarbeitet wird. Abb. 201

Anmerkung: Der Hinweis auf die Verarbeitungsform wird praktisch auf eine etwas andere Art und Weise berücksichtigt als in der Dateibeschreibung.

Um sich abschließend einen zusammenfassenden Überblick verschaffen zu können, sind die Verarbeitungsmöglichkeiten für die verschiedenen Speicherungsformen in Abb. 202 tabellarisch zusammengefaßt. Diese Aufstellung erhebt nicht den Anspruch auf umfassende Vollständigkeit. Von den hier genannten Grundformen für Verarbeitung und Speicherung lassen sich spezielle Erweiterungen ableiten. Diese sollen jedoch Gegenstand der Spezialliteratur bleiben.
Der Abb. 202 ist zu entnehmen, wie eine Datei bestimmter Speicherungsform verarbeitet werden kann. Mit Hilfe dieser Kenntnisse ist es verhältnismäßig einfach, im Rahmen eines komplexen Aufgabenbereiches die Dateiorganisation auf der Magnetplatte festzulegen. Dabei ist von vornherein zu berücksichtigen, welchen verschiedenen Programmläufen eine Datei später unterzogen werden soll. Aus der Klärung dieser Sachlage bestimmt sich dann die Speicherungsform eines Datenbestandes.
Eine Rechnungsschreibung wird im allgemeinen immer einen wahlfreien Zugriff zu den Artikel- und Kundensätzen voraussetzen. Auf Grund der Angaben in einer Bestellkarte (Abb. 194), einer Artikel- und einer Kundennummer, ist jeweils der zugehörige Datensatz einzulesen. Dadurch wird die für die Rechnungsschreibung benötigte Information im Hauptspeicher zur Verfügung gestellt.
Der Forderung nach wahlfreiem Zugriff kann jedoch entweder durch die gestreute oder logisch-fortlaufende Speicherungsform entsprochen werden. Im ersten Falle ist das Einlesen eines bestimmten Satzes über seine Satzadresse, im zweiten Falle über seinen Ordnungsbegriff möglich. Auf Grund besonderer Gesichtspunkte wird zwischen diesen beiden Möglichkeiten entschieden:

Dabei orientiert man sich

entweder an der günstigen Ausnutzung der Speicherkapazität,
oder an einer möglichst kurzen Zugriffszeit für einen Datensatz.

Beide Kriterien schließen oft einander aus.
Den vorangegangenen Ausführungen ist zu entnehmen, daß bei einem gestreuten Datenbestand, sofern keine 100 %ige Belegungsdichte vorliegt, freie Plätze vorhanden sind (Abb. 181). Bei einer 50 %igen Belegungsdichte wäre nur die Hälfte der vorhandenen Speicherkapazität in Anspruch genommen, die andere Hälfte bliebe ungenutzt.
Diesem Nachteil steht jedoch ein gewaltiger Vorteil gegenüber:
Auf Grund der direkten Adressierung ist bei Nennung der Satzadresse nur eine einzige Einstellbewegung des S-L-Kopfes nötig, um den gewünschten Satz schließlich in den Hauptspeicher einlesen zu können. Die dafür erforderliche Zeit heißt Zugriffszeit. Wenn die Einstellbewegung ca. 0,075 sec. und das Lesen eines 80 Bytes langen Satzes ca. 0,0005 sec. erfordert, beträgt die Zugriffszeit ca. 0,0755 sec.
Im Gegensatz dazu waren für das Einlesen eines logisch-fortlaufend gespeicherten Datensatzes insgesamt vier Einstellbewegungen des S-L-Kopfes (auf den Haupt-, Zylinder-, Spur-Index und den Datensatz selbst) erforderlich. Entsprechend ergibt sich für diese Zugriffszeit ein bedeutend höherer Wert.
Im Hinblick auf eine möglichst kurze Zugriffszeit ist also die gestreute der logisch-fortlaufenden Speicherungsform bei weitem überlegen.
Völlig andere Verhältnisse ergeben sich aber dann, wenn es um die bessere Ausnutzung der Speicherkapazität geht. Während bei der gestreuten Speicherungsform Satzlücken mit in Kauf genommen werden müssen, ist dies bei der des logisch-fortlaufenden Datenbestandes nicht mehr der Fall.

SPEICHERUNGSFORM \ VERARBEITUNGSFORM	starr fortlaufend	wahlweise	logisch fortlaufend
starr fortlaufend	möglich	X	X
wahlweise	möglich	möglich	X
logisch fortlaufend	möglich	möglich	möglich

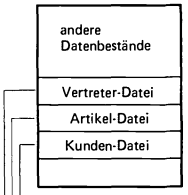
Verarbeitungsmöglichkeiten für die verschiedenen Speicherungsformen von Magnetplatten-Dateien (Die Normalfälle sind stark umrandet).

Abb. 202

Es muß also von Fall zu Fall entschieden werden, ob für eine Datei eine kurze Zugriffszeit oder die bestmöglichste Ausnutzung der Plattenkapazität ausschlaggebend sein soll. Für die Artikel-Datei (Abb. 181) wird die Entscheidung zugunsten der gestreuten Speicherungsform fallen, da der nicht belegte Anteil des Dateibereiches absolut gesehen gering ist. Selbst bei einer angenommenen 50 %igen Belegungsdichte beträgt er nach den Angaben in Abb. 181 nur einen Zylinder. Die Zugriffszeit für einen Artikelsatz beträgt nach den gemachten Ausführungen ca. 0,0755 sec.
Hätte man für die Artikeldatei die logisch-fortlaufende Speicherungsform gewählt, wäre zwar der dafür erforderliche Plattenbereich bedeutend kleiner ausgefallen (etwas mehr als 1 Zylinder, im Gegensatz zu den 2 Zylindern bei der gestreuten Speicherungsform), jedoch müßte dann eine höhere Zugriffszeit veranschlagt werden.

Die zunächst noch gering anmutenden Zeitunterschiede zwischen den Zugriffszeiten beider Speicherungsformen erlangen schnell an Bedeutung: Man braucht sich nur vorzustellen, daß im Laufe einer Rechnungsschreibung 15 000 mal Artikelsätze gelesen werden müssen. Dann werden aus Sekunden plötzlich Minuten und es ist durchaus ein wirtschaftlicher Vorteil, wenn ein Programm nicht erst nach 30, sondern bereits nach 25 min. abgeschlossen ist (Rechenzeit ist bares Geld). Dafür wird man gerne bereit sein, 1 Zylinder unbelegten Speicherplatzes bei der gestreuten Speicherungsform in Kauf zu nehmen.
Diese Überlegungen gelten im übertragenen Sinne natürlich auch für die Kunden-Datei. Hier ist man jedoch aus anderen Gründen gezwungen, die logisch-fortlaufende Speicherungsform zu wählen. Nach den Angaben in Abb. 187 umfaßt der Kundendatenbestand viele Tausende von Datensätzen. Sie sollen alle möglichst platzsparend auf der Platte untergebracht sein. Dafür ist ein großer Bereich erforderlich, den man natürlich gut ausgenutzt wissen möchte. Diese Forderung läßt sich mit Hilfe der logisch-fortlaufenden Speicherungsform erfüllen; bei ihr folgen die Sätze unmittelbar aufeinander. Es gibt keine Satzlücken wie bei der gestreuten Speicherungsform, es wird kein Speicherplatz „verschenkt". Eine einfache Abschätzung zeigt, wie groß dieser Anteil ungenutzten Speicherplatzes bei der gestreuten Speicherungsform sein könnte.
Für 4000 Kundensätze à 80 Bytes, deren Nummernkreis von 0001 bis 9999 reicht (Belegungsdichte 40 %), wäre im Falle der gestreuten Speicherungsform Platz für 9999 Sätze a 80 Bytes zu reservieren (40 Zylinder). Davon sind jedoch nur 40 % durch die vorhandenen Kundensätze kapazitätsmäßig ausgenutzt. Zusammengenommen bleiben also 24 Zylinder frei.
Sofern die Magnetplatte genügend Platz bietet, kann man über soviel Verschwendung ruhig einmal hinwegsehen. Doch auch eine Magnetplatte kostet Geld und schon aus diesem Grunde möchte man sie gut ausgenutzt wissen. Deshalb wird man für die Kunden-Datei die logisch-fortlaufende Speicherungsform wählen und in diesem Zusammenhang die höhere Zugriffszeit für einen Datensatz in Kauf nehmen. In allen solchen Fragen muß immer ein Kompromiß geschlossen werden. Letzten Endes entscheidet man sich für die „billigere" Möglichkeit. Sollte sich herausstellen, daß eingesparte Programmzeit auf Grund der kürzeren Zugriffszeit (bei der gestreuten Speicherungsform) mehr Einsparung bedeutet als eingesparter Speicherplatz (bei der logisch-fortlaufenden Speicherungsform), so dürfte die Entscheidung nicht schwerfallen. Es sei jedoch davor gewarnt, ein bestimmtes Programm gesondert zu betrachten.
Wirtschaftliche Erwägungen dürfen sich immer nur am Zusammenspiel aller Programme einer komplexen Organisation orientieren. Außerdem spielen bei der Festlegung der Speicherungsform für einen Datenbestand auch noch andere Überlegungen eine große Rolle. Schließlich wird eine Datei mitunter von vielen Programmen verarbeitet. So ist es bspws. denkbar, daß die Kunden-Datei auch zum Schreiben von Listen benutzt wird. Für eine solche Aufgabe allein gesehen wäre eine starr-fortlaufende Speicherungsform völlig ausreichend. Jedoch zwingt die Rechnungsschreibung zur logisch-fortlaufenden Speicherungsform der Kundendatei. Danach ist aber auch (Abb. 202) das Schreiben von Listen möglich, denn ein logisch-fortlaufend-gespeicherter Datenbestand kann auch starr-fortlaufend verarbeitet werden.
So wird man immer erst feststellen, welchen verschiedenen Programmläufen eine Datei unterzogen werden soll. Erst dann fällt die Entscheidung für eine zweckmäßige Speicherungsform.
Im Kapitel 20 ist bereits ausführlich auf den Begriff der Blockung bei Bandsätzen hingewiesen worden. Programmtechnisch war dabei die Festlegung eines Arbeits- und eines

Blockbereiches (für die Ein- bzw. Ausgabe) zu berücksichtigen. Mit jedem Lesebefehl wurde jeweils der nächste Datensatz aus dem Blockeinlesebereich in den Arbeitsbereich übertragen. Das Einlesen eines neuen Blockes erfolgte automatisch ohne zusätzlichen Programmbefehl.
Die Vorteile der Blockung, eine bessere Kapazitätsausnutzung des Datenträgers und ein beträchtlicher Zeitgewinn bei der Ausführung eines Programmes, bleiben jedoch nicht nur den Band-Dateien vorbehalten. Wie bereits erwähnt, läßt sich eine Blockung auch bei Plattensätzen durchführen. Das ist für jede Speicherungsform möglich. Grundsätzlich treten dabei keine wesentlich neuen Gesichtspunkte auf.
Die starr-fortlaufende Verarbeitung blockweise gespeicherter Datensätze erfolgt in Analogie zu der des Magnetbandes. Die logisch-fortlaufende Block-Verarbeitung setzt allerdings einen komplizierten Adreß-Index voraus, über den hier keine Aussagen erfolgen. Es ist jedoch gewährleistet, daß mit jedem Lesebefehl immer der logisch höhere Satz mit Arbeitsbereich zur Verfügung gestellt wird.
Die wahlfreie Block-Verarbeitung erscheint etwas problematisch. Der Lesebefehl bewirkt zwar das Einlesen des Blockes, der den gewünschten Datensatz enthält. Es bleibt aber jetzt dem Programmierer überlassen, diesen gewünschten Datensatz durch entsprechende Programmbefehle aus dem Blockeinlesebereich „herauszufischen" (zu lokalisieren) und in den Arbeitsbereich zu übertragen.



Satzbezeichnung	max. Anzahl der Sätze	Speicherungsform	Satzlänge	Erforderlicher Speicherplatz Anzahl der Zylinder + Spuren	Belegung von Zyl.	Spur	bis Zyl.	Spur
Vertreter-Satz	1000	starr-fortl.	80	4	128	0	131	9
Artikel-Satz	500	gestreut	80	2	132	0	133	9
Kunden-Satz	~ 4000	log.-fortl.	80	~ 22	134	7	155	9

Belegungsübersicht für die Vertreter-, Artikel- und Kunden-Datei eines Magnetplattenstapels.

Abb. 203

In Abb. 203 sind zur besseren Übersicht die in diesem Kapitel erwähnten Plattendateien schematisch zusammengestellt (Anordnung auf dem Magnetplattenstapel und Belegungsplan). Zur Programmverarbeitung ist dieser Magnetplattenstapel in eine Magnetplatteneinheit einzusetzen.

23. Die Programmbibliothek auf der Magnetplatte

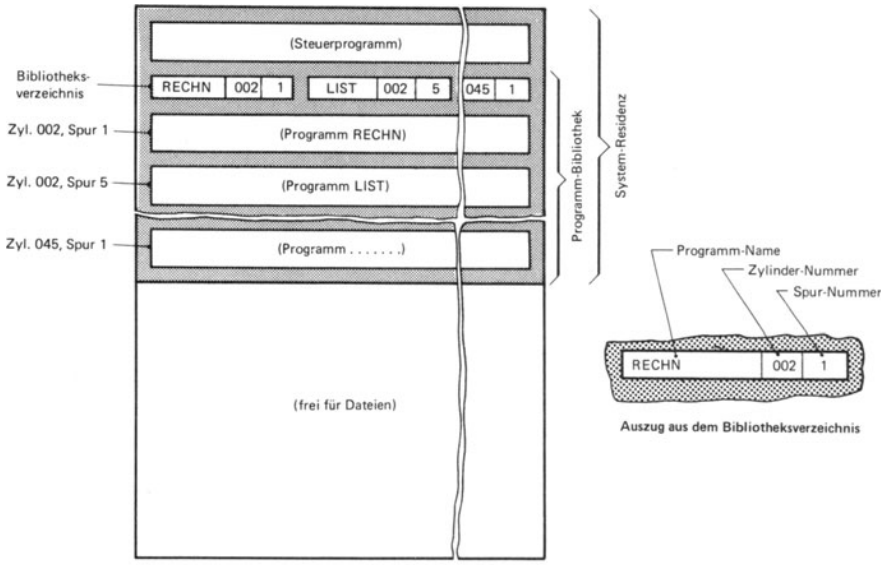
Die Entwicklung elektronischer Datenverarbeitungsanlagen erfolgt unter dem Gesichtspunkt zunehmender Leistungsfähigkeit. Dabei erstrebt man eine schnellere und funktionssichere Arbeitsweise. Unter diesem Aspekt kann auch das Laden eines Programmes in den Hauptspeicher verbessert werden. Bisher war man es nur gewohnt, „echte" Programm-Lochkarten über den Karten-Leser in den Hauptspeicher zu laden.
Hierbei sei nochmals ausdrücklich auf den Unterschied zwischen Programm- und Daten-Lochkarten hingewiesen: Für erstere steht der gesamte Hauptspeicher zur Verfügung; dort erscheinen nach dem Ladevorgang die einzelnen Programmbefehle und Definitionsangaben eng aneinandergereiht.
Das Einlesen von Datenlochkarten hingegen bleibt immer auf den Karteneinlesebereich begrenzt.
Das Laden eines Programmes erfordert eine bestimmte Zeit. Bei einer Leistungsfähigkeit des Karten-Lesers von 60 000 Karten/Stunde wären für 1 000 Programmlochkarten 60 sec. nötig.
In der Computertechnik sind 60 Sekunden jedoch schon eine „kleine Ewigkeit". Immerhin gibt es Rechner, die in einer Minute mehrere Millionen Rechenbefehle ausführen. Je schneller ein Programm geladen werden kann, um so besser ist der Computer zeitlich ausgenutzt (Zeit bedeutet Geld).
Um dieser Forderung zu genügen, müßte man „schnellere" Karten-Leser entwickeln. Diesem Streben ist jedoch aus konstruktiven Gründen eine Grenze gesetzt.
Einen Ausweg bietet die Verwendung externer Magnetspeicher (insbesondere Magnetband oder -platte). Von ihnen lassen sich Informationen in wesentlich kürzerer Zeit als von Lochkarten in den Hauptspeicher übertragen.
Voraussetzung ist natürlich, daß zuvor die Angaben der „echten" Programm-Lochkarten auf Magnetband oder -platte gespeichert wurden.
Von einer Magnetplatte[1] erfolgte das Einlesen von 1 000 Sätzen a 80 Bytes mit Programmbefehlen und Definitionsangaben in der Zeit von ca. einer halben Sekunde.
Beim Magnetband würde dieser Vorgang etwa eine Sekunde dauern[2]. Es mag noch nicht recht einleuchten, weshalb die „echten" Programm-Befehle zunächst auf einen externen Magnetspeicher übernommen und erst von dort aus in den Hauptspeicher geladen werden.
Dieser zusätzliche Aufwand erweist sich auch erst dann als vorteilhaft, wenn ein Programm mehrmals auszuführen ist. Eine Rechnungsschreibung z. B. läuft fast täglich. Folglich müssen nach der alten Methode die dazugehörigen „echten" Programm-Lochkarten (angenommen 1000 Stück) ca. 300 Mal im Jahr in den Hauptspeicher geladen werden. Insgesamt wären dafür also 300 Minuten = 5 Stunden aufzuwenden.
Ist dieses Rechnungsschreibungsprogramm auf einer Magnetplatte gespeichert, so sind für die gleiche Anzahl von Laderoutinen nur zweieinhalb Minuten erforderlich.
Computersysteme, bei denen das Programmladen von der Magnetplatte her erfolgt, arbeiten nach einem sogenannten Platten-Betriebssystem, während man beim Programmladen vom Magnetband her von einem Band-Betriebssystem spricht.

1 Lesegeschwindigkeit: 156 000 Bytes/sec. (angenommener Wert)
2 Lesegeschwindigkeit: 90 000 Bytes/sec. (angenommener Wert)

Der Begriff „Betriebssystem" birgt viele vorteilhafte Gesichtspunkte in sich. In diesem Zusammenhang werden nur jene von ihnen behandelt, die für das Verhältnis der hier gemachten Ausführungen notwendig sind.
Die folgenden Ausführungen beziehen sich auf Platten-Betriebssysteme. Die einzelnen Programme sind in einem ausgewiesenen Bereich der Magnetplatte, der sogenannte Programmbibliothek, gespeichert (Abb. 204).*
Die Frage, wie diese Programme dorthin gelangen, ist schnell geklärt:
Zur Kenntlichmachung erhält jedes Programm einen frei wählbaren Namen. Der Programmname wird in der ersten symbolischen Programmlochkarte nach formellen Gesichtspunkten aufgenommen (Abb. 205).

z. B.	Programm Rechnungsschreibung	:	RECHN
	Programm für das Auflisten von Lochkarten	:	LIST

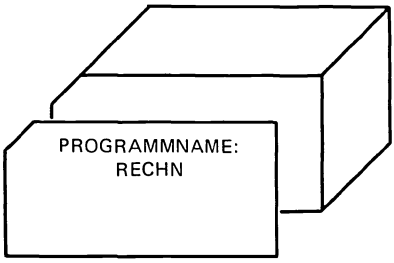
Nach dem Umwandlungslauf erscheint dieser unverändert in der ersten „echten" Programmlochkarte.



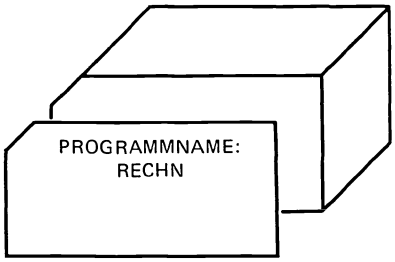
Magnetplattenstapel mit Systemresidenz

Abb. 204

* Die Festlegung der Lage und Größe der Programmbibliothek erfolgt nach bestimmten Richtlinien, über die hier nichts ausgesagt wird. Die Grenzen dieses Plattenbereiches sind so zu wählen, daß er später alle Programme aufnehmen kann.



symbolische



„echte" Programmlochkarten

Festlegung von Programmnamen

Abb. 205

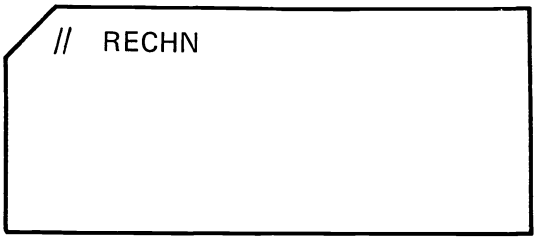
Jetzt gilt es, die Angaben dieser echten Programmlochkarten in die Programmbibliothek zu übertragen. Dieser Aufgabe dient ein sogenanntes BIBLIOTHEKS-Programm, welches die Herstellerfirma des Computers liefert.
Falls in die Programmbibliothek bereits andere Programme aufgenommen wurden (Abb. 204), ermittelt das BIBLIOTHEKS-Programm die Adresse des daran anschließenden freien Teiles. Von dieser Stelle ab werden die Befehle und Definitionsangaben des neuaufzunehmenden Programmes auf die Platte geschrieben.
Damit ist jedoch der Wirkungsbereich des BIBLIOTHEKS-Programmes noch nicht erschöpft; denn neben dem soeben geschilderten Ablauf wird zusätzlich ein sogenanntes Bibliotheksverzeichnis angelegt.
Dieses Bibliotheksverzeichnis ist im Grunde genommen nichts weiter als ein kleiner Adreß-Index. Seine einzelnen Angaben umfassen jeweils den Programm-Namen und die Plattenadresse, von der ab ein Programm in der Bibliothek gespeichert ist (Abb. 204).

Das Programm-Laden von der Magnetplatte

Angenommen, es bestünde die Aufgabe, eine Rechnungsschreibung auszuführen. Bevor es zu einer Verarbeitung der Bestell-Lochkarten kommen kann, muß das Programm RECHN von der Magnetplatte in den Hauptspeicher geladen werden. Dieser Vorgang wird durch eine sogenannte Steuerangabe eingeleitet. Eine solche Steuerangabe kann entweder durch Tastatur einer sogenannten System-Konsole oder über Steuerlochkarten dem Computer mitgeteilt werden. Aus Gründen der Anschaulichkeit sei hier auf „alte" Methode der Steuerlochkarten verwiesen.
Die Steuerlochkarten (Abb. 206) sind zu ihrer Kennzeichnung z. B. mit zwei Schrägstrich-Zeichen in den ersten beiden Spalten versehen, denen der Name des Programmes folgt.
Nach dem Einlesen dieser Steuerlochkarte stellt sich der S-L-Kopf auf das Bibliotheksverzeichnis ein. Sobald sich der Vermerk

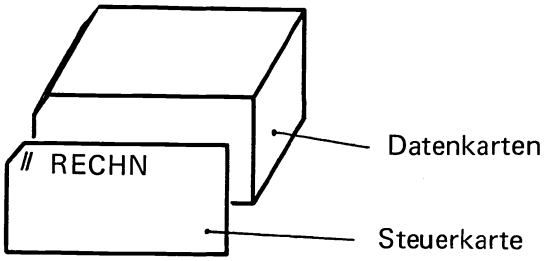
„RECHN 002 1"

unter ihm hindurchbewegt, stellt er automatisch auf Zyl. 002, Spur 1 ein, von wo an das Programm RECHN gespeichert ist. Anschließend werden die Befehle und Definitionsangaben dieses Programmes in den Hauptspeicher geladen. Erst dann kann das Einlesen und die Verarbeitung der Bestell-Lochkarten erfolgen (Abb. 207, 208).



Steuerkarte zum Laden des Programmes namens RECHN aus der Programmbibliothek der Systemresidenz in den Hauptspeicher

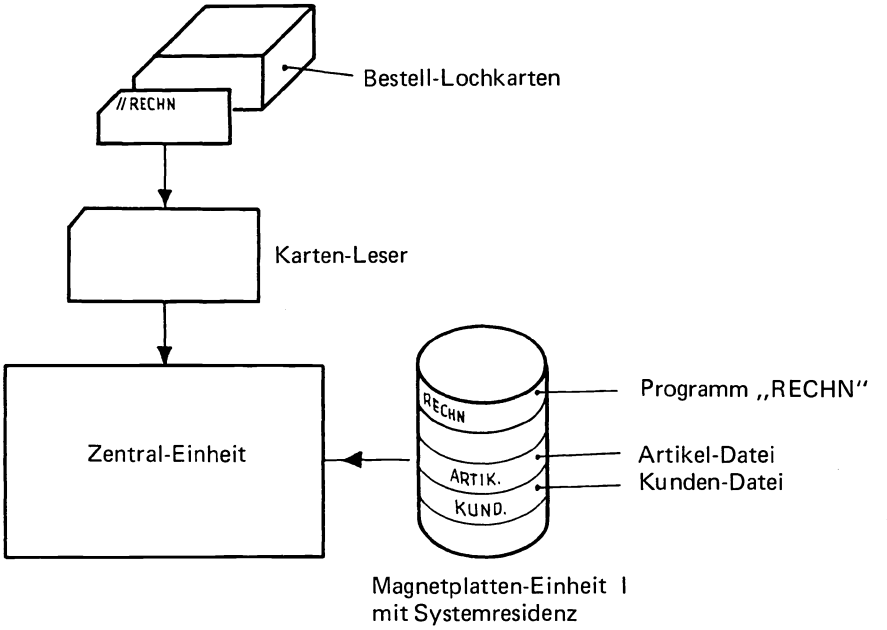
Abb. 206



Eingabefolge von Steuerkarte und Bestell-Lochkarten in den Karten-Leser für das Programm „RECHN"

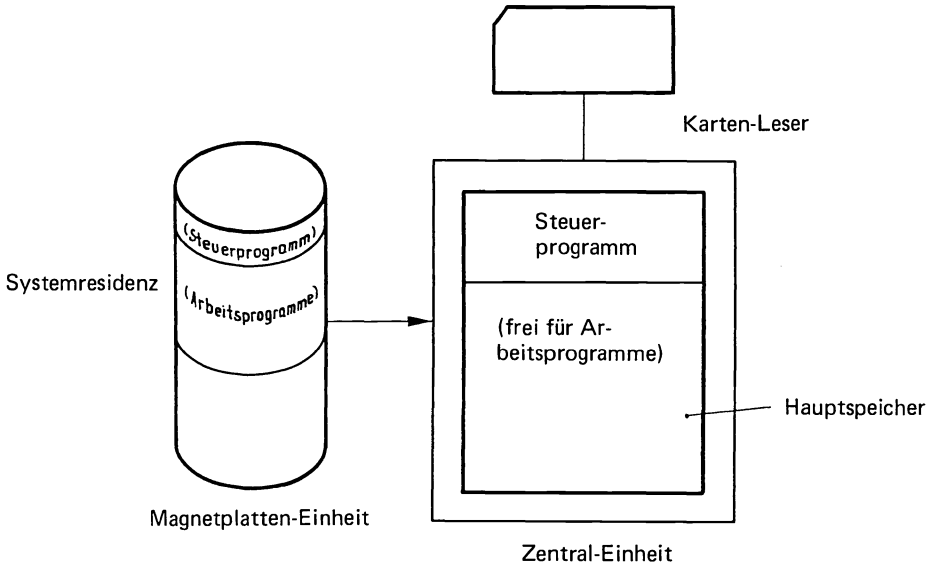
Abb. 207

Nach Abschluß der Rechnungsschreibung ist der Computer bereit, ein neues Programm auszuführen, welches wieder mittels einer entsprechenden Steuerlochkarte in den Hauptspeicher geladen werden kann.
Im Betriebssystem einer modernen Computeranlage finden sich „Aufpasser und Helfer". Sei arbeiten in Form eines Steuerprogrammes. Der Programmierer hat mit ihnen nichts zu tun. Sie werden bei Inbetriebnahme des Computers durch einen einfachen Knopfdruck von der Magnetplatte in einen besonderen Teil des Hauptspeichers geladen. Von dort aus steuern und überwachen sie den Ablauf von Arbeitsprogrammen, wie RECHN etc. Alle diese Arbeitsprogramme, zu denen auch später erwähnte Standardprogramme zählen, sind in die Programmbibliothek der Magnetplatte aufzunehmen. Sie können dann von dort aus mittels einer Steuerkarte in den Hauptspeicher geladen werden.
Die Standardprogramme werden ebenso wie das STEUER-Programm von der Herstellerfirma des Computers geliefert. Sie dienen der Durchführung häufig wiederkehrender Arbeiten (Programmumwandlung, Hauptspeicherinhalt ausdrucken, Aufnahme eines Programmes in die Programmbibliothek). Jedes Standardprogramm trägt einen besonderen Namen (wichtig für das Ausfüllen der Steuerkarte).



Laden des Programmes „RECHN" aus der Systemresidenz in den Hauptspeicher mit Hilfe einer Steuerkarte. Anschließende Ausführung der Rechnungsschreibung.

Abb. 208



Im Betriebssystem erfolgt die Ausführung jedes Arbeitsprogrammes unter Kontrolle des Steuerprogrammes

Abb. 209

Die Standardprogramme werden in

Verwaltungs-,
Dienst- und
Umwandlungsprogramme (Compiler)

eingeteilt. Folgend sind die wichtigsten von ihnen tabellarisch zusammengestellt und zum Teil auch erläutert.
Neben der Steuerkarte, die den Programmnamen enthält und dem Laden des Programmes in den Hauptspeicher dient, gibt es auch noch andere Steuerkarten, deren Bedeutung im Zusammenhang mit entsprechenden Beispielen noch erörtert wird.

	Standardprogramme
Verwaltungsprogramme	Bibliotheksprogramm
	etc.
Dienstprogramme	KARTE/PLATTE
	KARTE/DRUCKER
	KARTE/STANZER
	KARTE/BAND
	PLATTE/KARTE
	PLATTE/PLATTE
	PLATTE/DRUCKER
	PLATTE/BAND
	BAND/KARTE
	BAND/PLATTE
	BAND/DRUCKER
	BAND/BAND
	PLATTE LÖSCHEN
	HAUPTSPEICHERINHALT DRUCKEN
	SORTIEREN PLATTE
	SORTIEREN BAND
	MISCHEN PLATTE
	MISCHEN BAND
Umwandlungsprogramme	ASSEMBLER-COMPILER
	COBOL-COMPILER
	FORTRAN-COMPILER
	ALGOL-COMPILER
	PL/1-COMPILER

Sortieren mit Hilfe eines Standardprogrammes
Das Sortieren ist in der elektronischen Datenverarbeitung eine unentbehrliche Voraussetzung für eine zweckmäßige und wirkungsvolle Programmgestaltung (Kapitel 17).
Der Sortiervorgang wird nach dem kennzeichnenden Merkmal der Daten, dem Ordnungsbegriff, durchgeführt. Dieser kann numerischer oder alphabetischer Art sein und z. B. eine Artikelnummer, eine Kundennummer oder eine Ortsbezeichnung umfassen.
Sobald Lochkarten in eine vorgeschriebene Reihenfolge gebracht werden sollten, bewährte sich die Sortiermaschine als willkommener Helfer.

Nun ist es aber durchaus nicht so, daß Daten nur in Form von Lochkarten vorliegen können. Sie treten auch als Band- oder Plattensätze auf. Und in diesem Zusammenhang ist es auch denkbar, daß sich z. B. die Datensätze innerhalb einer Platten-Datei in ungeordneter Reihenfolge befinden.
Um sie sortieren zu können, mußte eine besondere Methode ersonnen werden, denn die Sortiermaschine versagt hier ihren Dienst: Plattensätze sind keine Lochkarten.
Es ist naheliegend, für eine solche Aufgabe den Computer heranzuziehen. Er kann durch ein

SORTIERPROGRAMM

angewiesen werden, die Datensätze einer Plattendatei in eine gewünschte Reihenfolge zu bringen, sie zu sortieren.
Ein derartiges Programm „SORTIEREN PLATTE" (SORT/PLATTE) wird von der Herstellerfirma des Computersystems geliefert. Sein Arbeitsablauf ist derart kompliziert, daß ein einfacher Programmierer überfordert wäre, es selbst aufzustellen. Aus diesem Grunde werden an dieser Stelle auch keine Einzelheiten über den Ablauf eines Sortierprogrammes behandelt. Es genügt, zu wissen, daß es ein solches Programm gibt und wie man damit arbeitet.
Dazu sind allerdings einige Vorbemerkungen erforderlich:
Aus den vorangegangenen Übungsfolgen ist bekannt, daß in der Dateibeschreibung eines Programmes u. a. folgende Angaben aufzunehmen sind:

Bezeichnung der externen Einheit (z. B. Kartenleser oder Platteneinheit I oder Platteneinheit II oder Bandeinheit I etc.)
Datei-Name
Ausdehnung des Datei-Bereiches innerhalb einer Unter- und Obergrenze*.

Die Herstellerfirma, die das „SORT/PLATTE"-Programm liefert, kann jedoch über diese Angaben keine Aussage treffen.
Schließlich richtet sich die Festlegung der Dateigrenzen immer nach der Zahl der anfallenden Sätze. Das wird jedoch von Fall zu Fall verschieden sein: Für 10 000 Vertretersätze à 100 Bytes ist ein größerer Platz auf der Platte erforderlich als für 800 Kundenumsatzsätze à 85 Bytes.
Außerdem muß es dem Benutzer der elektronischen Datenverarbeitungsanlage überlassen bleiben, in welche Platten-Einheit er einen Plattenstapel einlegen möchte. Er wird sich dabei von Gesichtspunkten leiten lassen, die einem zweckmäßigen Arbeitsablauf entsprechen.
Des weiteren liegt es auch einzig und allein in der Hand des Programmierers, den Dateinamen zu vergeben. Handelt es sich um Vertretersätze, wird als Dateiname vielleicht VERTR gewählt; bei Kundenumsatzsätzen hingegen wäre KUNUMS denkbar.
Auf Grund dieser Bemerkungen dürfte es also der Herstellerfirma nicht möglich sein, ein vollständiges SORT/PLATTE-Programm zur Verfügung zu stellen. In dem angelieferten symbolischen Programm würden jene Lochkarten fehlen, in denen die Dateiangaben über die externe Einheit, den Dateinamen und die Bereichsgrenzen enthalten sind. Das wäre jedoch kein „Beinbruch"; der Programmierer könnte sie je nach Anwendungsfall

* Diese Angabe ist nur bei Magnetplatten-Dateien erforderlich.

nachträglich einfügen. Anschließend müßte er dieses symbolische Programm einem Umwandlungslauf unterziehen, um ein arbeitsfähiges „echtes" Programm zu gewinnen (Kapitel 9).
Aus zeitlichen Gründen ist es jedoch nicht ratsam, diesen Weg zu beschreiten. (Ist es bspws. erwünscht, eine Vertreter-Datei zu sortieren, so müßte das symbolische „SORT/PLATTE"-Programm zunächst mit den fehlenden Dateiangaben vervollständigt und dann umgewandelt werden. Das auf diese Weise gewonnene echte „SORT/PLATTE"-Programm wäre nur für die Vertreter-Datei anwendbar).
Sobald die Sortierung einer anderen, z. B. einer Kundenumsatz-Datei gefordert wird, müßte wieder das symbolische „SORT/PLATTE"-Programm mit den entsprechenden Dateiangaben komplettiert und anschließend umgewandelt werden.
Um diesen enormen Zeitaufwand zu vermeiden (jede Umwandlung kostet Zeit), hat man eine besonders vorteilhafte Methode der Programmbenutzung entwickelt, die ein wiederholtes Umwandeln eines symbolischen Programmes bei wechselnden Dateiangaben überflüssig macht.
Dabei werden besondere

STEUERKARTEN (Lochkarten)

verwendet.
Unter Beachtung formeller Vorschriften ist es möglich, ein symbolisches Programm ohne (bestimmte, häufig wechselnde) Dateiangaben aufzustellen und umzuwandeln. Das so erhaltene (unvollständige) „echte" Programm bedarf dann aber vor seiner Ausführung jeweils einer entsprechenden Ergänzung. Diese Ergänzung erfolgt durch Steuerkarten, die die fehlenden Dateiangaben enthalten.
Die Steuerkarten werden dem „echten" Programm vorangestellt und mit diesem über den Karten-Leser in den Hauptspeicher eingelesen (Abb. 210).
Sämtliche Steuerkarten können z. B. durch Schrägstriche (//) in den Spalten 1 und 2 gekennzeichnet sein. Die Aufnahme der fehlenden Dateiangaben in die Steuerkarten erfolgt nach einem vorgeschriebenen Formalismus, über den hier nichts ausgesagt werden soll.
Die allgemein gehaltenen Angaben in den Steuerkarten (lt. Abb. 210) sind für das Verständnis ausreichend.
Verschiedene Programme erfordern je nach ihrem Aufgabenbereich eine oft unterschiedliche Vielzahl von Steuerkarten mit entsprechenden Angaben (derartige Einzelheiten sind in ganz speziellen Programmbroschüren vermerkt).
Für das „SORT/PLATTE"-Programm müssen demnach auch Steuerkarten mit Aussagen über den Ordnungs(Sortier)begriff vorgesehen werden. Es ist anzugeben, ob der Ordnungsbegriff numerisch oder alphabetisch ist, wieviel Stellen er umfaßt und ab welcher Stelle des Datensatzes er vermerkt ist (Abb. 210).
Damit sind fast alle Vorbereitungen getroffen — einem Sortierlauf der Vertreter-Datei steht nun nichts mehr im Wege: Die Steuerkarten werden samt dem „echten" Programmkartenpaket in den Hauptspeicher geladen. Der Computer nimmt seine Arbeit auf, bis alle (zuvor ungeordneten) Vertretersätze wohlgeordnet nach aufsteigender Vertreternummer auf der Magnetplatte stehen.
Für die komplette Abwicklung eines „SORT/PLATTE"-Programmlaufes ist das Vorhandensein eines sogenannten Arbeitsbereiches auf der Magnetplatte erforderlich. Seine Ausdehnung wird ebenfalls in einer Steuerkarte festgelegt (Abb. 210). Er dient dazu, die Vertretersätze vorübergehend aufzunehmen und wieder abzugeben, bis sie alle wohl

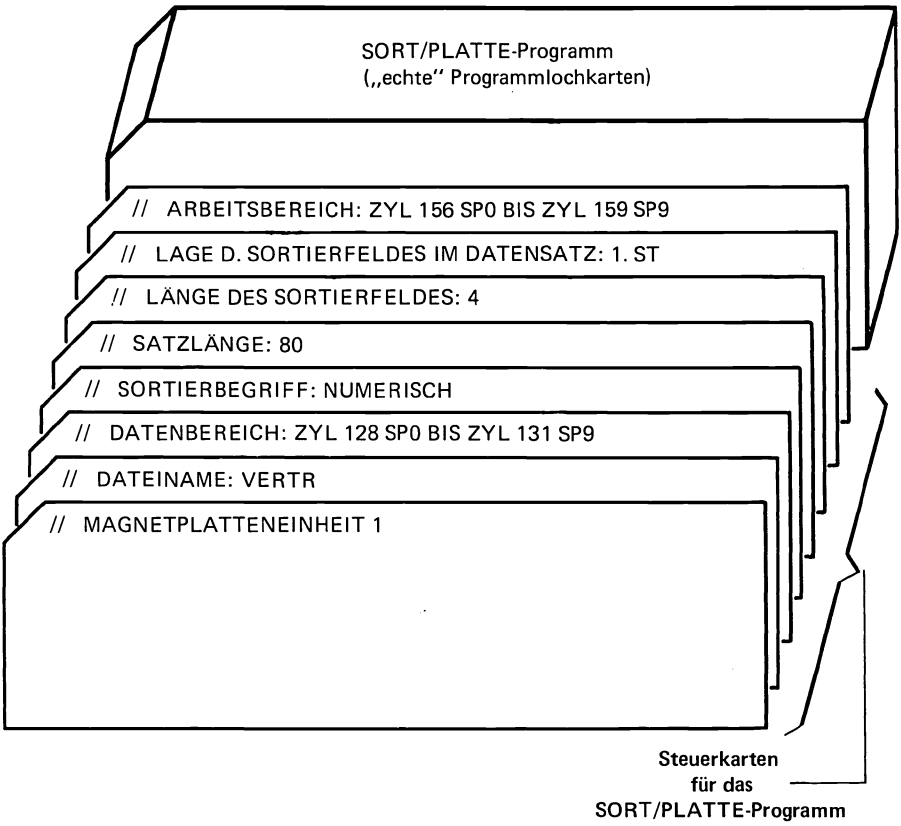


Abb. 210

geordnet in der Vertreter-Datei stehen. (Dieses „Hin und Her" erfolgt automatisch unter der Wirkung des „SORT/PLATTE"-Programms).
(Wollten Sie die Geldscheine Ihres Portemonnaies nach Beiträgen ordnen, so wäre das Vorhandensein eines Tisches o. ä. als Arbeitsplatz (Arbeitsbereich) zweckdienlich).
Die Verwendung eines „echten" Programmes in Verbindung mit Steuerkarten birgt einen großen Vorteil in sich. Nach der alten Methode wurde erst das symbolische Programm durch die jeweils fehlenden Dateiangaben ergänzt und dann umgewandelt.
Wechselnde Dateiangaben machten jeweils einen neuen Umwandlungslauf erforderlich (enormer Zeitaufwand). Ein „echtes" Programm (ohne spezielle Dateiangaben) kann hingegen für alle Dateien gleichermaßen benutzt werden; es ist von Fall zu Fall nur durch entsprechende Steuerkarten zu ergänzen.
Ebensogut wie Plattendateien können auch Banddateien sortiert werden. Diesem Zweck dient ein SORT/BAND-Programm. Es erfordert das Vorhandensein von mehreren Band-Einheiten. In eine dieser Band-Einheiten wird die Bandrolle mit den zu sortierenden Datensätzen, in die anderen werden leere Bandrollen eingelegt. Letztere übernehmen eine Funktion ähnlich der des Arbeitsbereiches auf der Magnetplatte.
Nach beendetem Programmlauf steht der sortierte Datenbestand auf einer dieser Magnetbandrollen zur Verfügung.

Neben den hier erwähnten Sortierprogrammen gibt es auch noch Mischprogramme für Platten- bzw. Bandsätze, so daß der Vorgang des Mischens nicht nur auf Lochkarten in Zusammenhang mit dem Kartenmischer (Kapitel 17) beschränkt bleibt.
Auch für andere, häufig wiederkehrende Arbeiten stehen entsprechende Standardprogramme zur Verfügung, z. B.:

KARTE/DRUCKER	:	Lochkarten auflisten
PLATTE/DRUCKER	:	Plattensätze auflisten (auf einer Liste ausdrucken)
BAND/DRUCKER	:	Bandsätze auflisten
KARTE/PLATTE	:	Den Inhalt von Lochkarten als Datensätze auf die Magnetplatte ausgeben
KARTE/BAND	:	Den Inhalt von Lochkarten als Datensätze auf ein Magnetband ausgeben. usw. usw.

Hierbei handelt es sich jeweils um eine Datenübertragung von einem Datenträger (KARTE, BAND, PLATTE) zu einem anderen. Der Programmierer braucht solche Programme nicht selbst zu erstellen, da sie zur Standardausrüstung eines Computersystems gehören und von der Herstellerfirma mitgeliefert werden.
Sobald sie in die Systemresidenz aufgenommen (katalogisiert) wurden, kann man in recht einfacher Weise durch Steuerkarten auf sie zurückgreifen.
Stellt man sich die Aufgabe, Vertreterlochkarten (Abb. 175) zu sortieren und nach aufsteigender Personalnummer aufzulisten, so sind zwei verschiedene Lösungswege möglich:

1. Die Vertreterlochkarten werden mit der Sortiermaschine sortiert. Das Programm „KARTE/DRUCKER" wird in den Hauptspeicher geladen und für die sortierten Vertreterlochkarten ausgeführt.
2. Hierbei bezieht man sich auf die in der Systemresidenz gespeicherten Standardprogramme.

In Abb. 211 ist gezeigt, in welcher Reihenfolge Steuer- und Datenkarten dem Karten-Leser zuzuführen sind.
In den Steuerkarten für das KARTE/PLATTE-Programm wäre u. a. der Plattenbereich anzugeben, in dem die unsortierten Vertreter-Sätze (sie stammen aus den unsortierten Vertreter-Lochkarten) aufzunehmen sind.
Mit Hilfe des SORT/PLATTE-Programmes erfolgt anschließend eine Sortierung der Vertreter-Sätze, die dann durch das PLATTE/DRUCKER-PROGR. ausgedruckt werden.
Die Standardprogramme können in Form „echter" Programmlochkarten von der Herstellerfirma des Computers bezogen werden. Es gibt zwei Möglichkeiten, mit diesen Standardprogrammen zu arbeiten:*

A. Programmladen über den Karten-Leser (Abb. 212)
B. Programmladen über der Systemresidenz (Abb. 213)

* Die Abb. 212 bzw. Abb. 213 beziehen sich auf die Sortierung eines Datenbestandes.

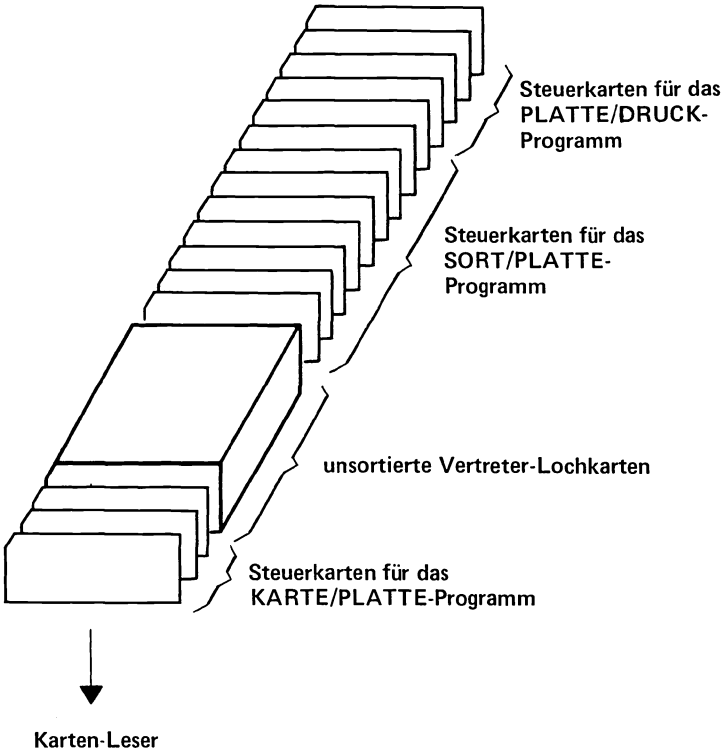
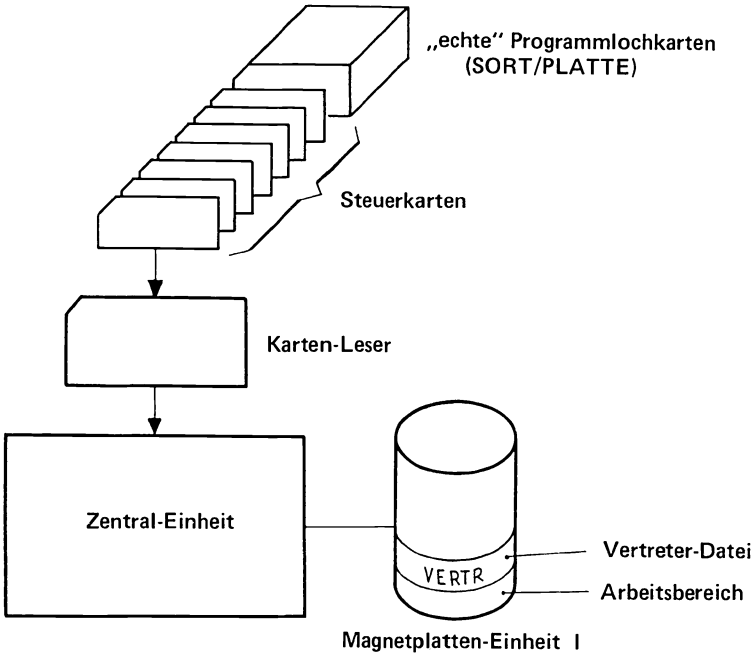


Abb. 211

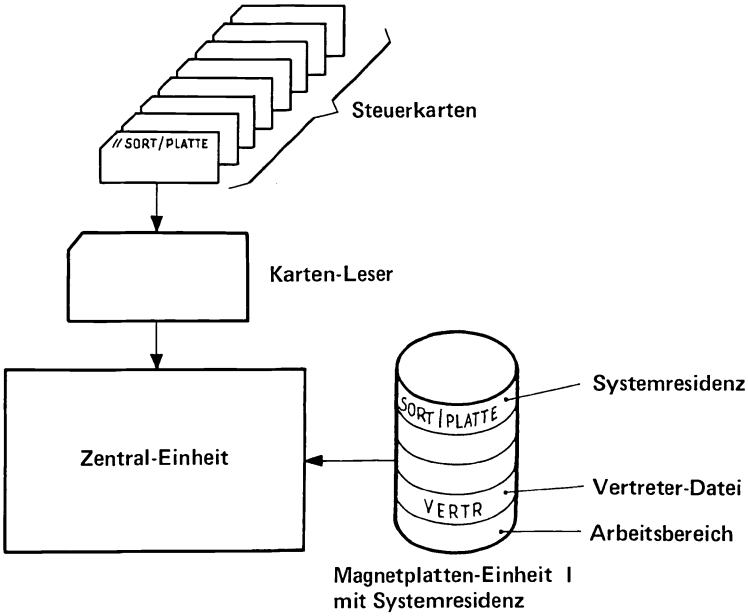
A. Die „echten" Programmlochkarten sind für den jeweiligen Anwendungsfall durch entsprechende Steuerkarten zu ergänzen. Das Programm wird in der in Abb. 212 angegebenen Reihenfolge über den Karten-Leser in den Hauptspeicher geladen. Der Computer ist dann bereit, die von ihm geforderte Arbeit auszuführen (z. B. das Sortieren eines Datenbestandes).

B. Das Laden des Standardprogrammes aus der Systemresidenz in den Kernspeicher wird durch eine Steuerkarte, die den Programmnamen enthält, ausgelöst. Für zusätzliche Angaben zur Vervollständigung des Programmes (bestimmte Dateiangaben etc.) dienen weitere Steuerkarten, die mit denen in Abb. 212 identisch sind.



Programmladen über den Karten-Leser

Abb. 212



Programmladen aus der Systemresidenz

Abb. 213

24. Die problemorientierten Programmiersprachen

Heutzutage gibt es eine große Zahl verschiedener Computertypen. Viele davon gestatten einen universellen Einsatz: sowohl für kommerzielle als auch für technisch-mathematische Probleme. Zur Programmierung stehen neben dem ASSEMBLER auch noch andere Programmiersprachen zur Verfügung, zum Beispiel FORTRAN oder ALGOL *) für technische Probleme und COBOL *) für kommerzielle Belange. PL/1 *) hingegen ist für beide Bereiche gleichermaßen geeignet.
FORTRAN, ALGOL, COBOL und PL/1 gehören im Gegensatz zu dem ASSEMBLER zu den problemorientierten Programmiersprachen. Sie kommen der menschlichen Ausdrucksweise sehr nahe: Entweder der Formelsprache der Techniker und Mathematiker oder dem „verständlichen Deutsch" der Kaufleute.
Ein Techniker benutzt zum Beispiel zur Berechnung des Volumens eines geraden Kreiskegels die Beziehung

$$V = 3{,}14 \cdot R \sqrt{(R^2 + H^2)}$$

(Es bedeuten: V = Volumen, R = Radius der Grundfläche, H = Höhe).
Ein Geschäftsmann hingegen hätte für einen Kunden mit vielen Wünschen aber wenig Geld keine Formel sondern etwa folgende Worte parat: „Wenn Sie Ihr Kreditlimit von DM 1 000 überziehen, kann ich leider keine weiteren Waren mehr an Sie ausliefern".
Für den Techniker wäre es von unschätzbarem Vorteil, könnte er sich in seinem FORTRAN-Programm zur Berechnung des Volumens des geraden Kreiskegels anstelle von vielen ASSEMBLER-Instruktionen nur auf eine einzige Anweisung (statement)

```
Volumen = 3,14 * R * SQRT (R**2 + H**2)
```

beziehen. (Volumen ist der symbolische Namen des Feldes, in dem das Ergebnis gebildet werden soll. R bezeichnet das Feld, in dem der Wert des Radius der Grundfläche gespeichert ist; H gilt für die Höhe. Ein * deutet ein einfaches Multiplikationszeichen an, während ein ** die nachfolgende Zahl zur Potenz erhebt. Die Bezeichnung SQRT (square route = Quadratwurzel) deutet ein Wurzelzeichen an).
Wie ließe sich aber nun bei einer programmierten Rechnungsschreibung das Problem lösen, ob ein Kunde sein Kreditlimit überzieht? Genau genommen, müßte vor dem Ausdrucken einer jeden Artikelzeile geprüft werden, ob mit der Fakturierung des betreffenden Artikels das Kreditlimit auch nicht überschritten wird. Andernfalls dürfte es nur zur einer Teillieferung kommen.
Eine entsprechende COBOL-Anweisung würde in diesem Fall etwa so lauten:
IF REST GREATER THAN KREDITLIMIT THEN GO TO TEILLIEF, OTHERWISE GO TO WEITER.
(REST und KREDITLIMIT sind symbolische Feldnamen. Das Feld REST enthält den Wert aller für eine Kundenrechnung bisher berücksichtigten Artikel, im Feld KREDITLIMIT ist die Höhe des Kreditlimits gespeichert. TEILLLIEF und WEITER sind sogenannte Paragraphennamen — vergleichbar mit ASSEMBLER-Instruktionsnamen — zu

*	FORTRAN	Formular Translating System (= Formel-Übersetzungs-System)
	ALGOL	Algorithmic Language (= algorithmische Sprache)
	COBOL	Common Business Oriented Language (= Allgemeine geschäftsorientierte Sprache)
	PL/1	Programming Language 1 (= Programmiersprache 1).

denen das Programm bei entsprechender Bedingung verzweigt. Der Paragraph WEITER müßte mit der Anweisung zum Ausdrucken einer Artikelzeile eingeleitet werden; unter TEILLIEF wären Anweisungen für eine Teillieferung auszuführen).
Die Tatsache, daß COBOL in den Vereinigten Staaten von Amerika entwickelt wurde, erklärt das Auftreten englischer Schlüsselworte (unterstrichen) in den Programmanweisungen. Wie diese Schlüsselworte zu handhaben sind, ist in besonderen Vorschriften festgelegt. Symbolische Feld- und Paragraphennamen darf der Programmierer frei wählen, so wie er es vom ASSEMBLER her gewohnt ist.
Zum besseren Verständnis der zuvor aufgeführten COBOL-Anweisung sei hier die deutsche Übersetzung der englischen Schlüsselworte mit angegeben:

IF REST GREATER THAN KREDITLIMIT THEN GO TO TEILLIEF, OTHERWISE GO TO WEITER.

WENN REST GRÖSSER ALS KREDITLIMIT, DANN GEHE NACH TEILLIEF, ANDERNFALLS GEHE NACH WEITER.

Sämtliche problemorientierten Programmiersprachen lassen sich verhältnismäßig leicht erlernen. Sie sind sehr aussagefähig und verständlich. Mit ihrer Hilfe kann ein Programm in bedeutend kürzerer Zeit erstellt werden, als es mit Hilfe des ASSEMBLERS möglich wäre.
Bei allen diesen Vorteilen scheint natürlich die Existenzberechtigung des ASSEMBLERS in Frage gestellt. Und doch hat es seinen Grund, weshalb gerade in diesem Buch dem ASSEMBLER eine so große Bedeutung beigemessen wurde. Denn nur so war es überhaupt möglich, eine fundierte Vorstellung von der programmtechnischen Arbeitsweise des Computers zu erlangen, die ebenso wie der ASSEMBLER auf dem sogenannten Zwei-Adreß-Prinzip beruht.
Bei allen Operationen, bei Übertragungen, Additionen etc., werden zumeist zwei Kernspeicherfelder miteinander verknüpft. In einer symbolischen wie auch in einer echten Instruktion sind die Adressen dieser Hauptspeicherfelder durch die beiden Operanden bezeichnet. Aus dieser Tatsache leitet sich der Begriff Zwei-Adreß-Prinzip ab.
Nach den Ausführungen im Kapitel 15 ermitteln sich die Operanden echter Maschinenbefehle aus der Angabe von Registern und Displacements. Auf diese Kenntnisse ist ein Programmierer des öfteren angewiesen; zum Beispiel wenn er ergründen muß, weshalb der Computer die Ausführung einer bestimmten Instruktion und damit die Fortführung des Programmes versagt. Die Ursachen einer solchen Unterbrechung können vielfältiger Natur sein: Entweder stehen in Zahlenfeldern alphabetische Angaben (mit denen der Computer natürlich nicht rechnen kann) oder in einem Befehl wird auf eine Adresse (in einem Register) verwiesen, die es in Wirklichkeit gar nicht gibt, die über die Größe des Kernspeichers hinausgeht. Die Reihe derartiger Beispiele ließe sich beliebig fortsetzen.
Um die Ursachen von diesen Programmunterbrechungen zu ergründen, gibt es einen recht wirksamen Weg. Man veranlaßt das Ausdrucken des gesamten Hauptspeicherinhalts. So erhält der Programmierer einen genauen Überblick über alle echten Instruktionen und den Inhalt der einzelnen Kernspeicherfelder.
Auf diese Informationen kann er sich dann bei der Fehlersuche stützen, was ihm bei entsprechenden ASSEMBLER-Kenntnissen recht leicht fallen dürfte. Selbst ein COBOL- oder FORTRAN-Programmierer sollte allein schon aus diesen Gründen auf Kenntnisse in der Zwei-Adreß-Programmierung nicht verzichten.
Unabhängig davon ist der ASSEMBLER besonders dazu geeignet, die programmtechnische Arbeitsweise des Computers zu erklären. Mit Hilfe von COBOL oder FORTRAN etc. wäre so etwas nicht möglich gewesen, denn dann hätte man lediglich über rein formelle Programmiervorschriften sprechen können, ohne jegliches Hauptspeicherbezoge-

nes Denken.
Egal, ob man sich später mit FORTRAN oder PL/1 befassen möchte, entsprechende ASSEMBLER-Kenntnisse werden einem immer zugute kommen. Vieles ist dann plötzlich selbstverständlich geworden, wenn auch die Dateien, Definitionen und Instruktionen in jeder Programmiersprache einer anderen Schreibweise genügen müssen.
Es würde natürlich den Rahmen dieses Buches sprengen, sollte hier im einzelnen darauf eingegangen werden. Doch auf eines sei noch kurz hingewiesen, daß es nämlich für jede symbolische Programmiersprache einen eigenen Compiler gibt. So sind zum Beispiel ASSEMBLER-Programme stets mit einem ASSEMBLER-Compiler und COBOL-Programme stets mit einem COBOL-Compiler umzuwandeln.

25. Computergenerationen und Betriebssysteme

Der Begriff der Generation wurde nun auch aus dem rein menschlichen Bereich auf das Gebiet der elektronischen Datenverarbeitung übertragen: Man spricht von einer ersten, zweiten, dritten Computergeneration und munkelt bereits von einer vierten.
Jede Computergeneration ist durch ganz bestimmte Merkmale gekennzeichnet, die sich wie Meilensteine in der Entwicklung der Elektronik ausnehmen.
1946 wurden die ursprünglich mit mechanischen Relais bestückten Rechenanlagen von den Vertretern der ersten Computergeneration abgelöst. Diese war bereits mit Elektronenröhren (Bildtafel 2) ausgestattet; (elektro-)mechanisch bewegte Teile, wie die Relais, konnten entfallen. Damit waren die ersten, rein elektronisch arbeitenden Rechner geboren. Als deren erstes Modell wurde der sogenannte „ENIAC" gebaut. Dieser enthielt unter anderem ca. 18 000 Elektronenröhren, was mit einem enormen Heizleistungsbedarf verbunden war; ganz zu schweigen von der durch die große Erwärmung bedingten Kühlanlage.
Welch ein Jubel muß deshalb erst unter den Computer-Konstrukteuren geherrscht haben, als der Transistor (Bildtafel 2) erfunden wurde. Der Transistor wies gegenüber der Elektronenröhre bedeutende Vorzüge auf: Einen kleineren Raumbedarf, einen geringeren Betriebsstrom, kürzere Schaltzeiten und eine ungleich höhere Lebensdauer. Die zweite Computergeneration machte sich diese Vorteile zunutze. 1954 konnte das Modell „TRADIC" fertiggestellt werden, für das ausschließlich Transistoren und andere Halbleiterbauelemente *) Verwendung fanden.
Seit dieser Zeit ist die Elektronenröhre als elektronisches Bauelement weitgehend aus der Rechnerentwicklung verbannt — die Vorzüge des Transistors setzten sich durch.
In der Folgezeit machte die Entwicklung der Elektronik weitere Fortschritte. Man war ohnehin schon immer bestrebt, die Leitungen zwischen den einzelnen Bauelementen möglichst kurz zu halten. Denn je kürzer die Leitung war, umso schneller konnte diese auch von einem Impuls durchlaufen werden und umso schneller konnte der Computer rechnen.
Unter diesem Gesichtspunkt wurden dann Mikrobauelemente und integrierte Schaltkreise entwickelt (siehe Bildtafeln).
In den integrierten Schaltkreisen sind die einzelnen Bauelemente und deren Verbindungsleitungen innerhalb eines äußerst kleinen Siliziumplättchens enthalten. Der Herstellungsprozeß dafür ist äußerst kompliziert; er umfaßt Aufdampf-, Druck-, Ätz- und Diffusionsvorgänge. Um eine Vorstellung von den Größenordnungen in der integrierten Technik zu bekommen, sei erwähnt, daß sich auf einem einzigen Siliziumplättchen von wenigen Quadratmillimetern Größe Tausende von Transistoren, Dioden und Widerstände integrieren lassen.

* Halbleiterbauelemente
sind zum Beispiel elektrische Stromrichter, Transistoren, Photozellen und Dioden. Sie bestehen aus Halbleitermaterial. Darunter versteht man einen Stoff, dessen elektrische Leitfähigkeit zwischen der eines Metalles und eines Isolators liegt. Diese elektrische Leitfähigkeit nimmt mit steigender Temperatur oder bei geringen chemischen Zusätzen (Fremdatome) stark zu. Durch eine gezielte Dotierung (z. B. Zugabe von Arsen- oder Indiumatomen) zum Halbleiter-Grundmaterial (z. B. Germanium) lassen sich sogenannte p- oder n-Leiter erzeugen, bei denen die Stromleitung durch quasi positive oder negative Ladungen erfolgt.
Durch eine geschickte Kombination n- und p-leitender Schichten lassen sich die Halbleiterbauelemente realisieren.

Wollte man jetzt die dritte Computergeneration kennzeichnen, so müßten neben den Mikrobauelementen und integrierten Schaltkreisen auch die sogenannten Betriebssysteme Erwähnung finden.
Um es kurz vorwegzunehmen, ein Betriebssystem dient der Steuerung und Überwachung der Arbeit einer modernen Datenverarbeitungsanlage. Es besteht aus einer Vielzahl von Systemprogrammen. Diese sind entweder auf einer Magnetplatte oder einem Magnetband gespeichert. Im ersten Falle spricht man von einem plattenorientierten, im zweiten Falle von einem bandorientierten Betriebssystem.
Die folgenden Ausführungen beschränken sich auf einen der wesentlichsten Bestandteile eines Betriebssystemes – auf das Organisationsprogramm. Ein einfacher Knopfdruck genügt, um das Organisationsprogramm innerhalb kürzester Zeit von der Magnetplatte (bzw. dem Magnetband) in einen Spezialbereich des Hauptspeichers zu laden.
Erst dann kann die eigentliche Arbeit des Computers unter der Regie des Organisationsprogrammes beginnen. Insgesamt erfüllt das Organisationsprogramm drei Funktionen: Die Funktion der Steuerung des Ablaufteiles, die Funktion der Daten-Ein/Ausgabe und die Funktion des Monitors.
Erstere umschließt unter anderem die Steuerung des Ladevorganges eines Benutzerprogrammes (zum Beispiel Rechnungsschreibung), bei dem jede Instruktion im Hauptspeicher einen Platz zugewiesen bekommt, außerdem die Analyse von Programmunterbrechungen, die Steuerung der Bedienungseinrichtungen und vor allem die Steuerung der für eine EDV-Anlage gewählten Betriebsart*).
Die Steuerung der Daten-Ein/Ausgabe bezieht sich größtenteils auf die Ausführung der Schreib- und Lesebefehle. Wenn in einer Dateibeschreibung für Plattensätze eine Länge von 100 Bytes angegeben wurde, so hat das Organisationsprogramm in Verbindung mit der Ein/Ausgabe-Steuerung dafür zu sorgen, daß auch wirklich nur 100-stellige Plattensätze eingelesen bzw. ausgegeben werden.
Der Monitor hingegen ermöglicht den aufeinanderfolgenden Ablauf mehrerer Programme, ohne daß der Operator für jedes Programm die Lade- und Start-Taste wiederholt betätigen muß.
Eine ganz besondere Bedeutung innerhalb eines Betriebssystemes kommt dem Laden von echten (Benutzer-)Programmen in den Hauptspeicher zu.
Das Einlesen von echten Programmlochkarten über den Karten-Leser (Kapitel 9) erforderte noch relativ viel Zeit (mehrere Sekunden). Diese Sekunden würden sich allerdings zu beträchtlichen Werten addieren, wenn ein Programm mehrere hundert Male geladen (und ausgeführt) werden soll.
Gelänge es, den Ladevorgang zu verkürzen, so könnte der Computer umso schneller mit der Ausführung des entsprechenden Programmes beginnen. Eingesparte Zeit bedeutet auch hier eingespartes Geld. Und aus diesem Grunde ist man bei Betriebssystemen dazu übergegangen, auch die Benutzerprogramme auf Magnetplatte oder -band zu speichern. Von dort aus können sie innerhalb von Sekundenbruchteilen in den Hauptspeicher geladen werden, viel schneller als es per Programm-Lochkarten über den Karten-Leser möglich wäre. (Die Anweisung, welches Programm geladen werden soll, nimmt der Computer in Form einer einzelnen Steuer-Lochkarte oder über eine entsprechende Nummernwahl entgegen).
Diese einfache Möglichkeit des Programmladens ist für viele der nachstehend aufgeführten Betriebsarten von EDV-Anlagen Voraussetzung.

* Nähere Erklärungen dazu auf den folgenden Seiten

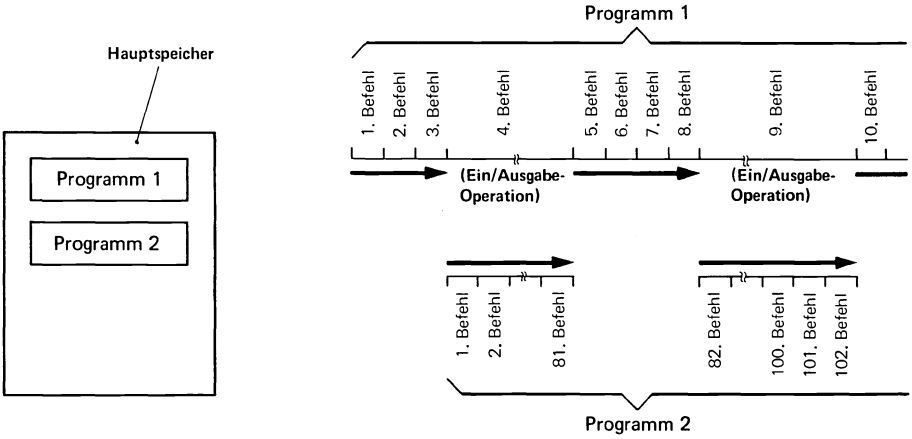
Der Stapelbetrieb

Diese Betriebsart wird heute in den meisten Fällen angewandt. Für jedes auszuführende Programm ist zunächst eine größere Menge von Daten zu sammeln (zum Beispiel eine größere Anzahl von Auftrags-Karten für eine Rechnungsschreibung), welche dann in einem „Schub" verarbeitet werden. Es wäre sehr unwirtschaftlich, wollte man wegen einer einzigen Auftrags-Karte den Computer zum Schreiben einer einzigen Rechnung veranlassen.
Der Begriff Stapel ließe sich leicht aus der Vorstellung ableiten, daß die Steuer- und Daten-Lochkarten für ein bzw. mehrere Programme zunächst als Stapel in das Magazin des Karten-Lesers eingeführt werden müssen.
Mit der Weiterentwicklung der Dateneingabemöglichkeiten in den Computer hat seit geraumer Zeit die Lochkarte eine große Konkurrenz erhalten, die sie immer mehr verdrängt.
Die Daten werden nicht mehr wie bisher abgelocht, sondern in einer selbständigen „Station" über eine Tastatur auf handliche kleine, flexible Magnetplatten (Disketten) oder Magnetbänder aufgezeichnet. Durch ihre hohe Aufzeichnungsdichte kann z. B. eine Diskette ca. 2000 Lochkarten ersetzen. Doch der Vorteil der Diskette liegt nicht allein in der Platzersparnis, sondern in der sehr kurzen Zeit, in der ihre Daten in einer entsprechenden Eingabe-Einheit in den Computer eingelesen werden können. Während für das Einlesen von 2000 Lochkarten noch ca. 2 min. aufzubringen waren, sind für das Einlesen der Daten einer Diskette lediglich wenige Sekunden erforderlich.
Mit Hilfe der Stapel-Betriebsart werden ein Großteil aller Abrechnungs- und Planungsaufgaben sowie technisch-wissenschaftliche Berechnungen abgewickelt. Ganz im Gegensatz dazu steht der

Der Realzeitbetrieb

Hier müssen die Programme zur Verarbeitung plötzlich anfallender Daten stets betriebsbereit sein. Diese Forderung ist zum Beispiel bei Abfrage- und Prozeßsteuerungssystemen erfüllt:
Die Polizei, die sich per Programm aus einer Verbrecher-Datei sofort alle rotäugigen Betrüger mit einer Körpergröße über 1,90 m und einem Schmiß auf der linken Wange heraussuchen lassen möchte, wird sicherlich ungern sehr lange auf die Antwort warten.
Als weiteres Beispiel seien die Platzbuchungssysteme moderner Fluggesellschaften erwähnt, bei denen in Großraumspeichern Angaben über die Platzverteilung in den Flugzeugen enthalten sind. Bildschirmgeräte und Fernschreiber in den verschiedenen Büros der Fluggesellschaft stehen über Telefonleitungen mit dem Computer in der Zentrale in Verbindung. Der vom Kunden gewünschte Flug braucht nur eingetastet zu werden und bereits nach kurzer Zeit liegt die Antwort vor, ob in der gewünschten Maschine noch Plätze frei sind. Dann kann die Platzbuchung durch eine weitere Tastatureingabe vorgenommen werden – der Computer reduziert daraufhin die Anzahl der noch verfügbaren freien Sitzplätze für das entsprechende Flugzeug in der Datei des Großraumspeichers. (Oftmals sind mit einer solchen automatischen Buchung auch das Ausdrucken der Flugkarten und die Zusammenstellung von Passagierlisten verbunden).
Prozeßrechner stellen die höchste Anforderung an den Realzeitbetrieb. So müssen beispielsweise bei der Steuerung von Weltraumflügen innerhalb von Sekundenbruchteilen Kurskorrekturen für die Rakete ermittelt werden.

Um einen Computer besonders gut auszunutzen, wird in vielen Betriebssystemen von der Möglichkeit Gebrauch gemacht, mehrere Programme quasi gleichzeitig ablaufen zu lassen. Auf diese Art und Weise vermeidet man Wartezeiten, die bei Ablauf eines einzigen Programmes auftreten würden.
Diese Wartezeichen sind durch die relativ langen Ausführungszeiten von Schreib/Lesebefehlen bedingt. Während das Einlesen einer Lochkarte beispielsweise ca. 50 Millisekunden dauert, erfordert die Durchführung einer einfachen Addition manchmal nur 0,00001 Millisekunden. Während eines Schreib/Lesebefehles arbeitet wohl die entsprechende Ein/Ausgabe-Einheit, nicht aber die Zentral-Einheit. Sie muß warten, bis der angeforderte Datensatz im Ein/Ausgabe-Bereich des Hauptspeichers steht; erst dann können die im Programm folgenden relativ „schnellen" Übertragungs-, Vergleichs- oder arithmetischen Befehle zur Ausführung gebracht werden.
Während der Wartezeit bliebe die Zentral-Einheit also ungenutzt. Um diese Zeitverschwendung (Rechenzeit ist bares Geld!) zu vermeiden, wird beim Multiprogramming eine zeitlich verzahnte Ausführung mehrerer Programme vorgesehen. Das Prinzip des Multiprogrammings ist ganz einfach: Im Hauptspeicher sind mehrere Programme vorhanden (Abb. 214). Wird an einer Stelle des Programmes I ein Ein/Ausgabe-Befehl erteilt, so gelangen während der damit verbundenen relativ langen Wartezeit Befehle des Programmes II zur Ausführung. Nach Ablauf der Wartezeit wird dann das Programm I wieder fortgesetzt. So werden beide Programme zeitlich verzahnt ausgeführt. Es entstehen keine Wartezeiten mehr, die Zentraleinheit ist ständig beschäftigt.



(die stark ausgezogene Linie deutet den zeitlichen Verlauf für die abwechselnde Ausführung von Befehlen der Programme 1 und 2 an).

Schematische Darstellung des Multiprogramming-Prinzipes

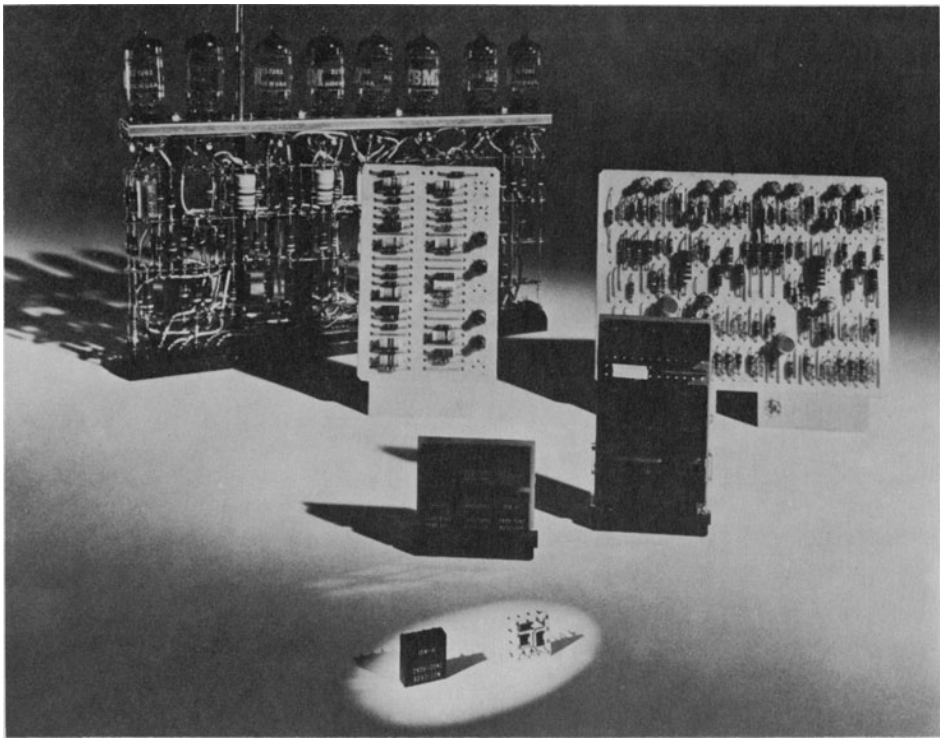
Abb. 214

Je nach dem Komfort eines Betriebssystems gibt es verschiedene Stufen des Multiprogrammings, die sich durch die Anzahl der quasi gleichzeitig ablaufenden Programme (oft mehr als 2 Stück) unterscheiden.

Das Timesharing ist ein Verfahren, bei dem eine Datenverarbeitungsanlage quasi gleichzeitig von mehreren Benutzern in Anspruch genommen wird. Das Organisationsprogramm übernimmt die Zeitaufteilung für die Ausführung der verschiedenen Programme: Zwei Sekunden Programmzeit für den Benutzer 1, drei Sekunden für den Benutzer 2, eine Sekunde für den Benutzer 3 und wiederum zwei Sekunden für den Benutzer 1 usw. Je nach Festlegung wird die Gesamtzeit in einzelne Intervalle aufgeteilt, die den Benutzern zyklisch oder entsprechend einer bestimmten Priorität zur Verfügung stehen. Oftmals sind die Benutzer über weit entfernt liegende Datenendstationen mit der Zentral-Einheit verbunden. Von den Datenendstationen können Daten und Programme eingegeben und die errechneten Ergebnisse entgegengenommen werden. Jeder Benutzer verkehrt mit der Anlage so, als stünde diese nur für ihn allein zur Verfügung. Er merkt gar nicht, daß der Computer sekundenweise auch für andere Teilnehmer rechnet.
Das Multiprocessing gestattet ganz im Gegensatz zum Multiprogramming die wirklich gleichzeitige Ausführung mehrere Programme. Anlagen für einen solchen Betrieb setzen allerdings zwei oder mehr Zentraleinheiten voraus, von denen jede ihre eigene Programmsteuerung besitzen muß.
Verschiedene Betriebssysteme berücksichtigen das Prinzip eines virtuellen Speichers (virtuell = scheinbar). Dies stellt für den Benutzer eine wesentliche Erleichterung dar. Denn nun kann der Computer auch ein Programm ausführen, das größer ist als der zu seiner Speicherung erforderliche Hauptspeicherplatz. Aus Kapitel 6 ist noch bekannt, daß ein Programm nur dann ausgeführt werden kann, wenn es im Hauptspeicher verfügbar ist. Was aber, wenn der Hauptspeicher zur Aufnahme des Programmes zu klein ist? Die Lösung dieses Problems ist einfach: Das Programm wird in einzelne Segmente (pages = Seiten) aufgeteilt. Ein Teil dieser Segmente wird im Hauptspeicher, der Rest auf einem externen Speicher, z. B. einer Magnetplatte untergebracht. Die Programmausführung bleibt natürlich — wie könnte es auch anders sein — auf den Hauptspeicher beschränkt. Dort sind jedoch nur einige Programmsegmente vorhanden. Wird zur weiteren Programmausführung ein anderes Segment benötigt, das sich noch auf der Magnetplatte befindet, so muß dieses in Blitzesschnelle in den Hauptspeicher geladen werden. Im Austausch dafür (der Platz im Hauptspeicher ist „knapp") muß ein nicht mehr benötigtes Programmsegment aus dem Hauptspeicher auf die Magnetplatte ausgelagert werden. Das ständige Aus- und Einlagern der Programmsegmente erfolgt völlig automatisch unter Steuerung des Betriebssystemes. Der Benutzer des Computers braucht sich also um diese Vorgänge nicht im entferntesten zu kümmern. Er kann so programmieren, als stünde ihm zur Ausführung seines umfangreichen Programmes ein ausreichend großer Hauptspeicher zur Verfügung; auch wenn wir wissen, daß dies nur scheinbar der Fall ist, daß der „große" Hauptspeicher nur scheinbar (virtuell) existiert.
Nach allen diesen Ausführungen beginnt der Computer förmlich zu leben — alle seine Bauteile, die sogenannte Hardware, bilden mit dem unter dem Begriff Software zusammengefaßten System- und Benutzerprogrammen ein funktionstüchtiges System.
Die Welt, in der wir heute leben, ist ohne Computer nicht mehr denkbar. Ein kleiner Hinweis, der in der Presse erschien („Handelsblatt") kann dies nur unterstreichen. Danach waren im Jahre 1977 in der Bundesrepublik Deutschland über 125 000 Computer mit einem Wert von über 32 Milliarden DM installiert. Und der Computer setzt seinen Siegeszug weiter fort.

BILDANHANG

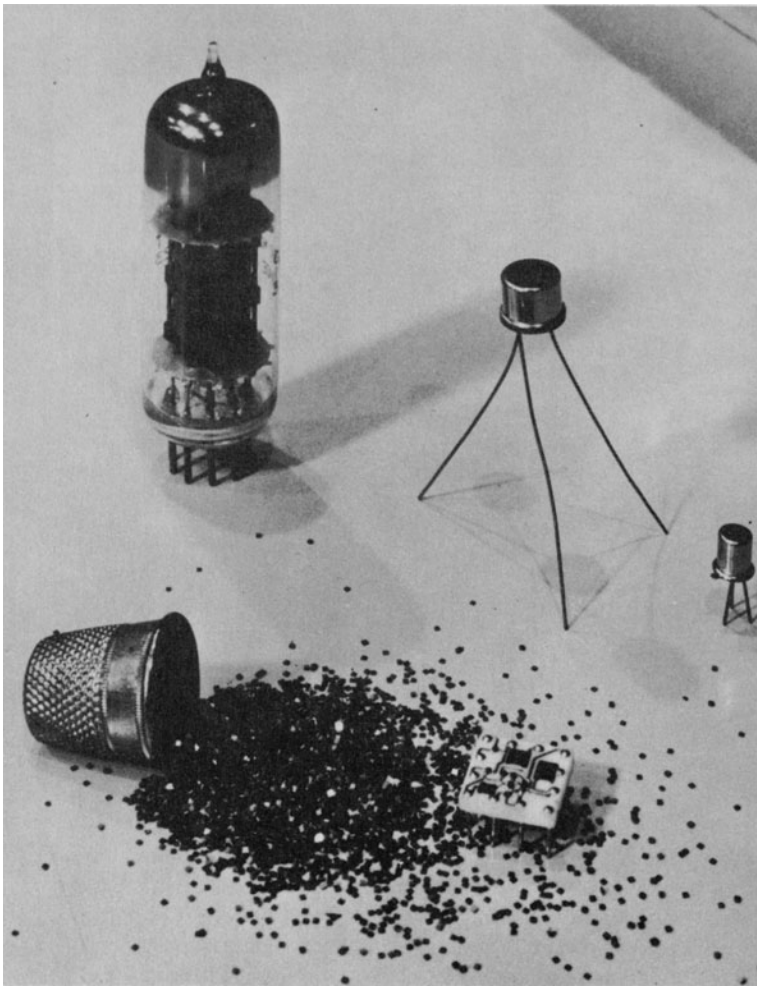
mit Abbildungen aus der Technologie der verschiedenen Computergenerationen und mit Ansichten moderner EDV-Anlagen und -Geräte verschiedener Herstellerfirmen.



Dieses Bild veranschaulicht ausgezeichnet die Technologie der drei Generationen von Elektronenrechnern. Im Hintergrund sind einige Schaltkreise der ersten mit Elektronenröhren bestückten Rechner zu sehen. Davor eine sogenannte SMS-Karte (gedruckte Schaltungen) der zweiten Generation und als Vertreter der jüngsten Generation eine Trägerplatte, auf die 12 Mikroschaltungen – die jede nur 11,5 x 11,5 mm groß sind – aufgelötet sind.

IBM-Pressefoto
360/010

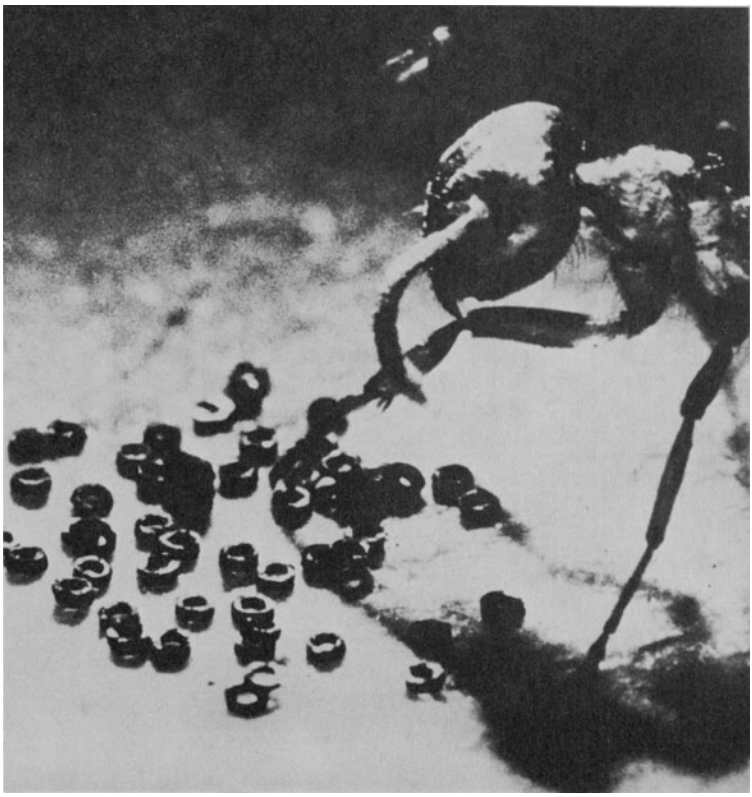
Tafel 1



Je größer und leistungsfähiger elektronische Datenverarbeitungsanlagen werden, desto kleiner müssen ihre einzelnen Bauelemente werden. Die Entwicklung elektronischer Datenverarbeitungsanlagen begann bei den Relais und Elektronenröhren und setzte sich fort bis zu den Transistoren und Dioden. In der „dritten Generation der elektronischen Datenverarbeitung" beim IBM System/360 wurde der nächste Schritt, der Schritt zu den Mikrobauelementen verwirklicht. Bei dieser neuen Technologie SLT Solit Logic Technology — oder Mikro-Schalttechnik sind die verwendeten Transistoren nicht größer als Salzkörner.

IBM-Pressefoto
B 108

Tafel 2

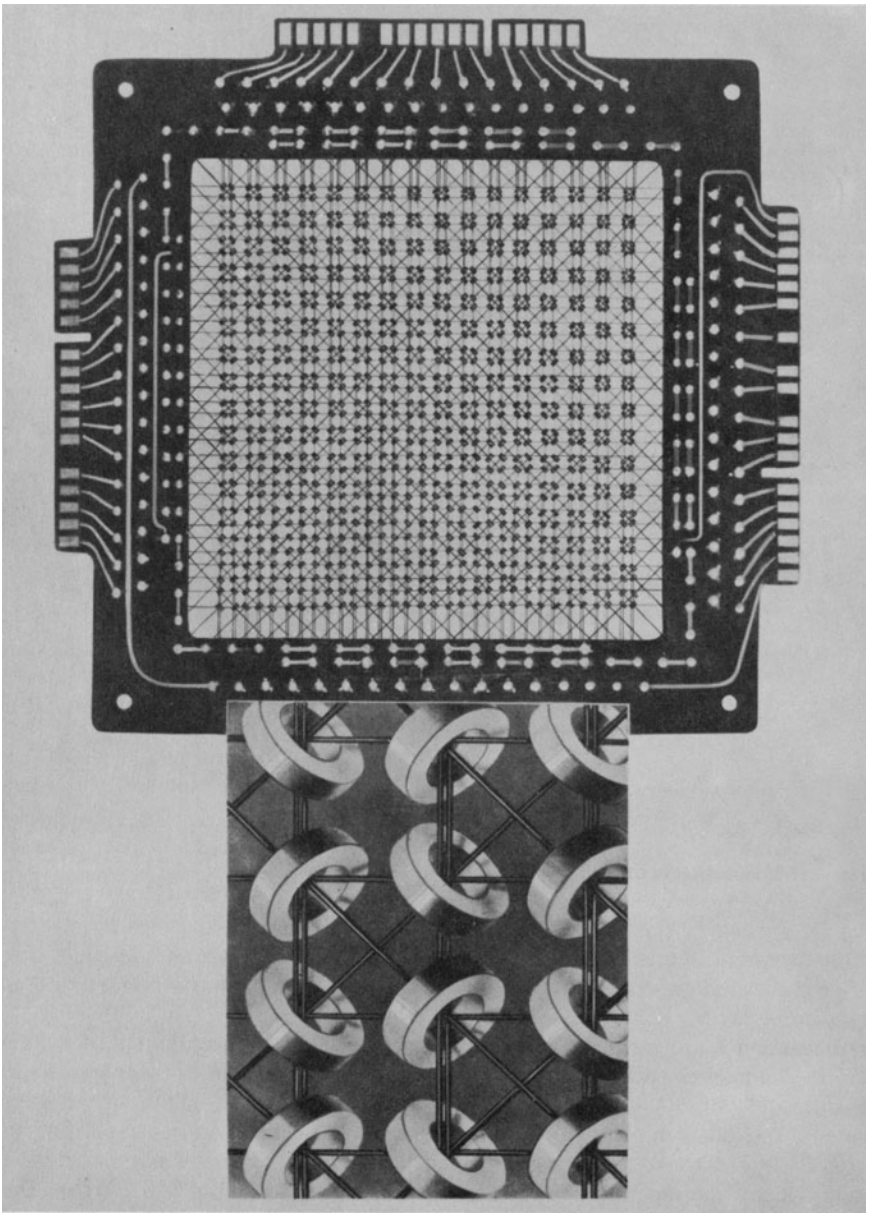


Extrem kleine Magnetkernringe

Spielzeug für Ameisen? Fast so sehen die extrem kleinen Magnetkernringe für elektronische Datenverarbeitungsanlagen aus. Nur 0,2 Millimeter Durchmesser haben die neuartigen Speicherelemente – so klein, daß nicht einmal der Fühler einer Ameise hindurch gezwängt werden kann.
Hergestellt werden diese kleinen Einheiten für die Speicherung nach einem sehr alten Prinzip: Die Methode der Kerzenmacher wurde wieder ausgegraben und der modernen Technik nutzbar gemacht. Wo die landläufige Fertigung versagte, halfen sich Wissenschaftler und Techniker mit einem mittelalterlichen Trick: Ein Nylon-Faden wird mehrmals durch Bäder aus Lack und Magnetpulver gezogen. Diese „Magnet-Kerze" wird dann von ihrem „Docht" befreit und in einem Spezialverfahren in kleinste Teile zersägt. Das Ergebnis: Äußerst kleine Magnetringe, die mit geringer Leistung geschaltet werden können. Die Speicherkapazität des Computers kann auf diese Weise beträchtlich gesteigert werden.

IBM-Pressefoto
260167

Tafel 3



Magnetkernspeicher—Ebene

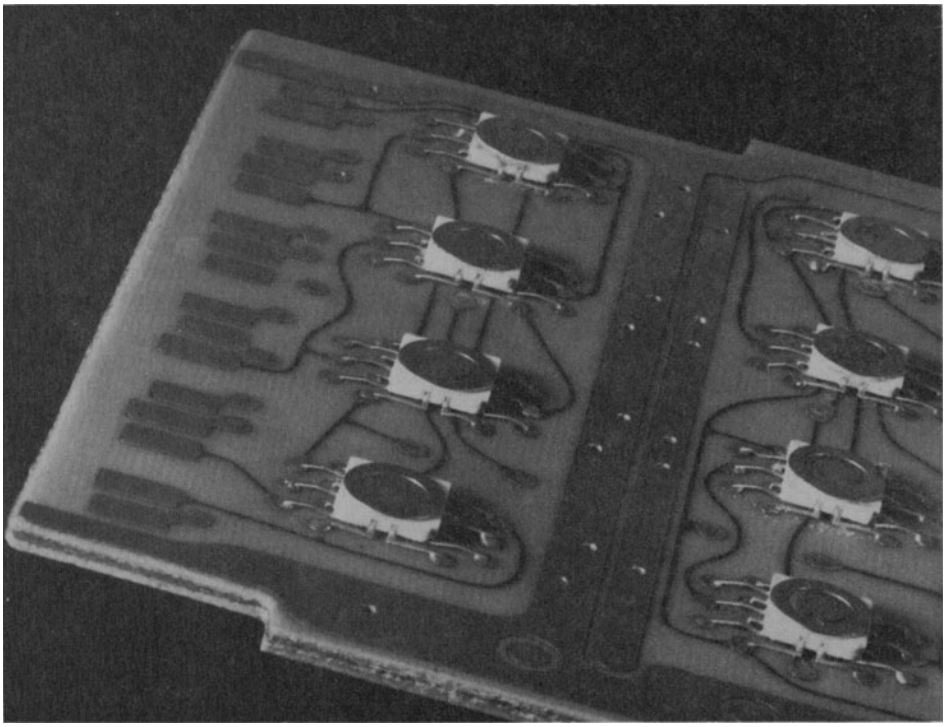
(UNIVAC-Foto)

Tafel 4

In ihrer dritten Computer-Generation, dem IBM System/360, benutzt die IBM eine von ihr entwickelte Festkörpertechnik. Hier werden die einzelnen Produktionsstufen der SLT-Mikrobaugruppen (SLT = Solid Logic Technology) gezeigt:
Das gepresste und gestanzte Keramik-Plättchen (1) wird mit metallisierter Tinte bedruckt (2). Die Widerstände werden ebenfalls aufgedruckt (3). Beim nächsten Schritt (4) werden 12 Kupferstifte, die als Verbindungen dienen, angebracht und mit den gedruckten und verzinnten Leitungen (5) verbunden. Anschließend werden die Widerstände mit einem winzigen Sandstrahlgebläse auf ihre benötigte Kapazität „zurechtgestutzt" (6). Im vorletzten Arbeitsgang werden Transistoren und Dioden an ihren Platz automatisch angebracht (7). Schließlich wird die ganze Mikrobaugruppe mit einer Schutzschicht überzogen (8).

IBM-Pressefoto

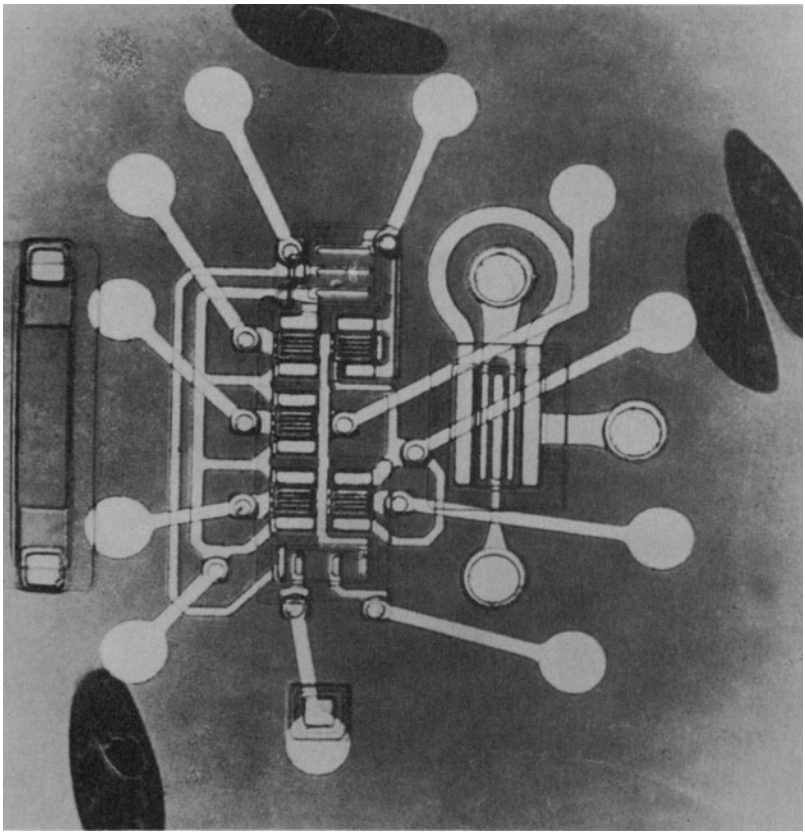
Tafel 5



Mehrlagige gedruckte Schaltung

(Siemens Pressebild)

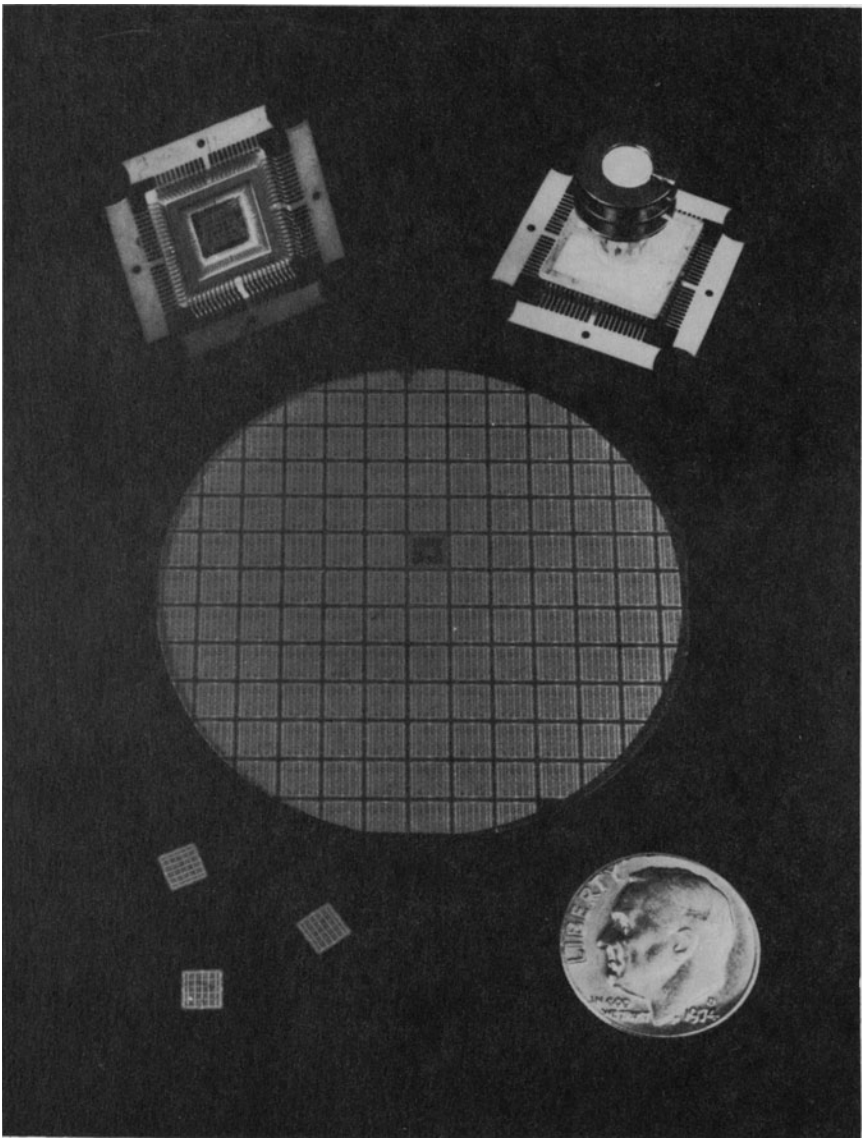
Tafel 6



Paramecia heißen die kleinen Einzeller an der Peripherie des Bildes. Sie erscheinen groß, verglichen mit den Ausmaßen dieses neuen monolithischen Schaltkreises, der von IBM entwickelt wurde. Es handelt sich um den bisher kleinsten und schnellsten Monolithen. Die Schaltzeit beträgt nur noch 400 Picosekunden (billionstel Sekunden). Die gestreiften Gebilde in der Bildmitte sind Transistoren. Sie sind sogar noch kleiner als der Kern eines der abgebildeten Einzeller. Zum Beispiel ist der Emitter eines solchen Transistors nur noch ein fünfhundertstel Millimeter breit.

IBM-Pressefoto

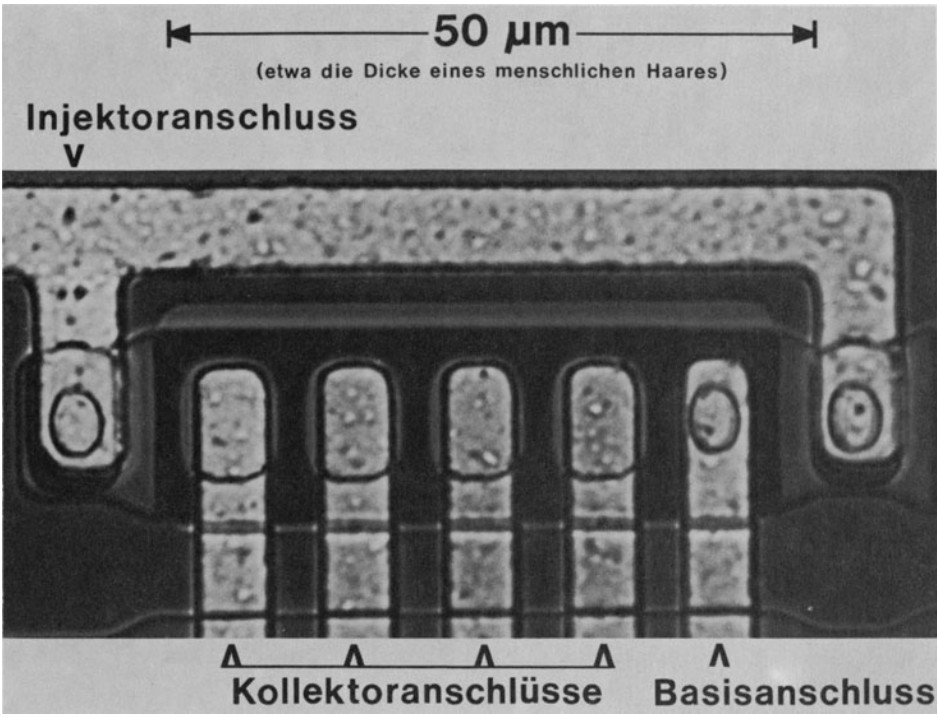
Tafel 7



Darstellung eines Amdahl-LSI-Chips (LSI = large scale inegrat'.) Das Silizium-Plättchen kann bis zu 100 Chips umfassen. Ein Chip besteht aus 11 Schichten, hat eine Fläche von ca. 0,9 cm^2, ist in einer Keramik-Packung montiert und mit 84 Außenanschlüssen versehen. Zur Luftkühlung ist es mit Kühllamellen (Bild oben) ausgestattet.

Amdahl–Pressefoto

Tafel 8

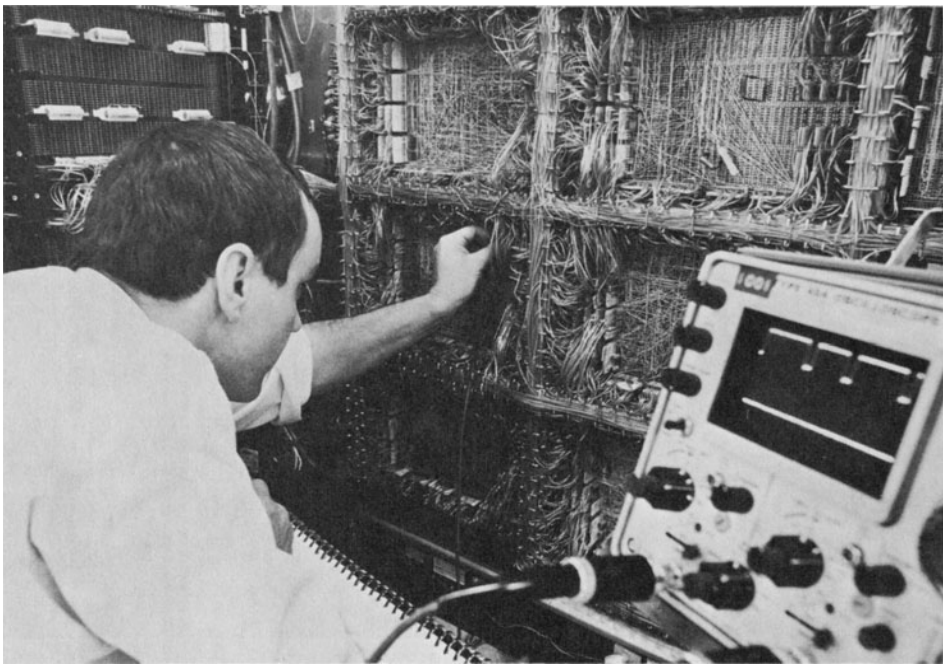


MTL-Grundbauelement:

ein Multikollektortransistor mit 4 Kollektoranschlüssen.

IBM Pressefoto
180377/5A

Tafel 9

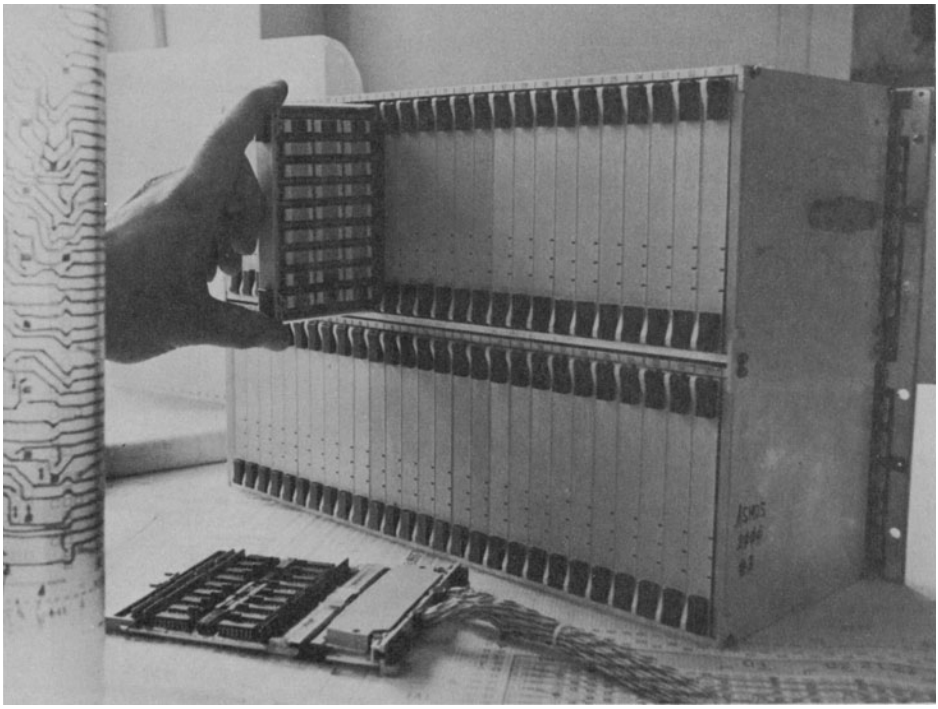


Produktion Werk Mainz IBM System/370 Modell 145

Prüfung einer Zentralen Recheneinheit der Systemfamilie / 370—145 in Mainz.
Nach dem Aufbau der Zentralen Recheneinheit werden alle Funktionen dieser Maschinen geprüft. Ähnlich wie ein Arzt mit seinem Hörrohr die Diagnose stellt, wird hier mit einem Oszillograph die rechte Impulsfolge einer Maschine ausgeprüft.

IBM Pressefoto
151271

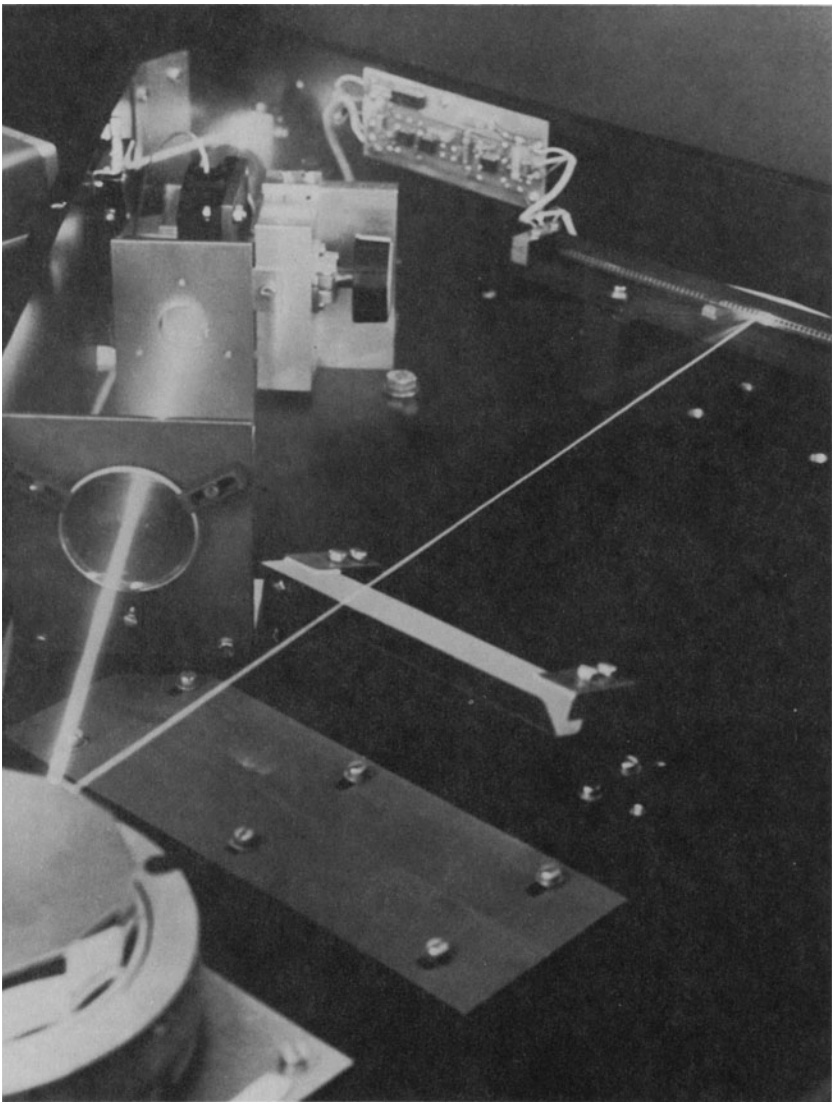
Tafel 10



Die neuen Halbleiterspeicher ASMOS 2000 von Siemens haben eine maximale Speicherkapazität von zwei Millionen Byte. Sie sind mit modernster Halbleitertechnik, und zwar mit dynamischen MOS-Bausteinen von 16 KBit, ausgerüstet.

siemens pressebild

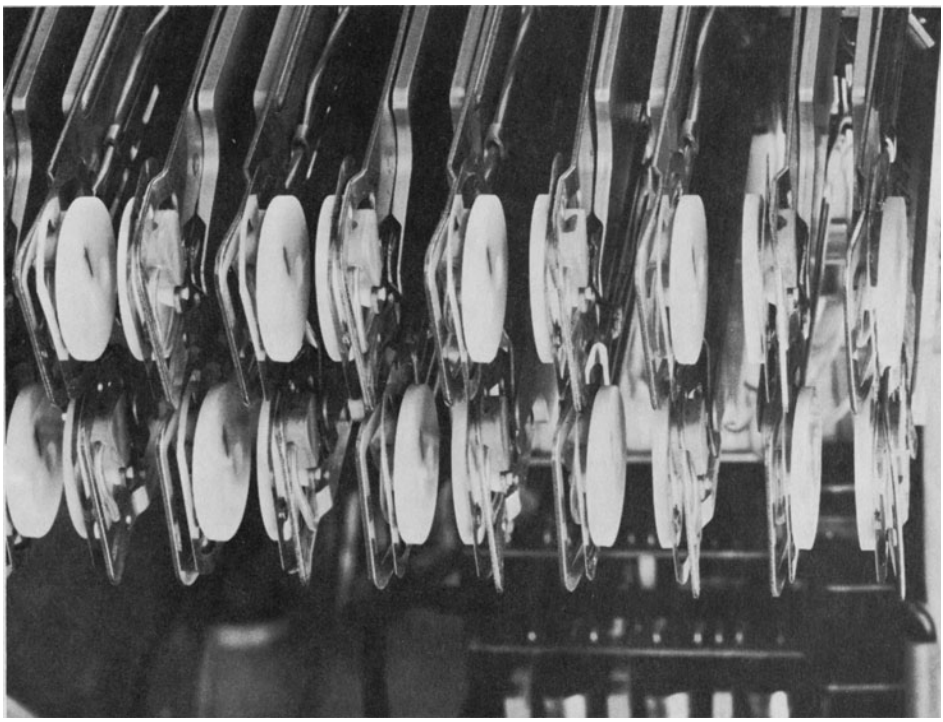
Tafel 11



Beim neuen Laserdrucker 3352 von Siemens werden die einzelnen Druckzeichen nicht mehr von Typen, sondern mit Hilfe von Laserstrahlen erzeugt. Damit erreicht man Druckgeschwindigkeiten von maximal 1,2 Millionen Zeilen pro Stunde oder rund 70 000 Zeichen pro Sekunde. Das ist etwa das zehnfache, was konventionelle Schnelldrucker schaffen.

siemens pressebild

Tafel 12

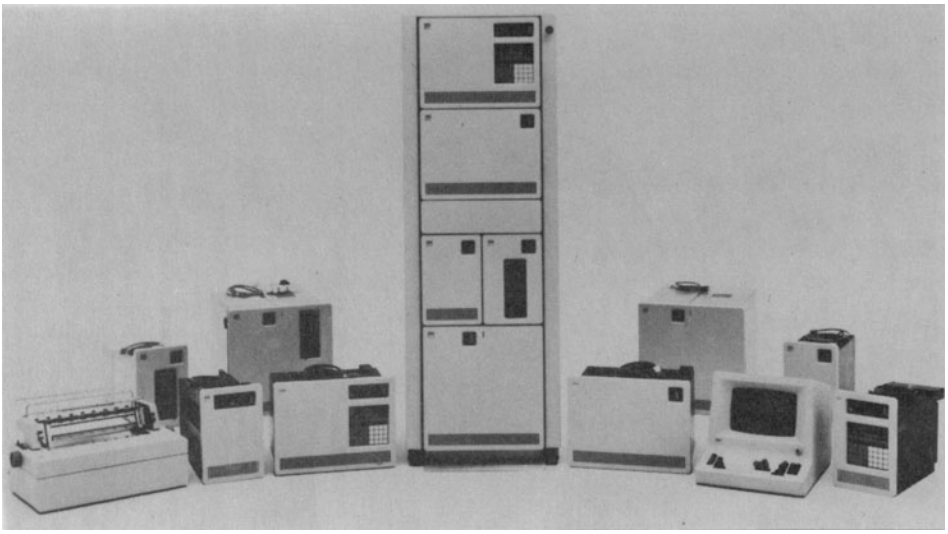


Produktion Werk Mainz

Magnetplattenspeicher IBM 3330. Die Magnetköpfe haben die Aufgabe, die Informationen von den Magnetplatten zu speichern und gespeicherte Informationen zu lesen.

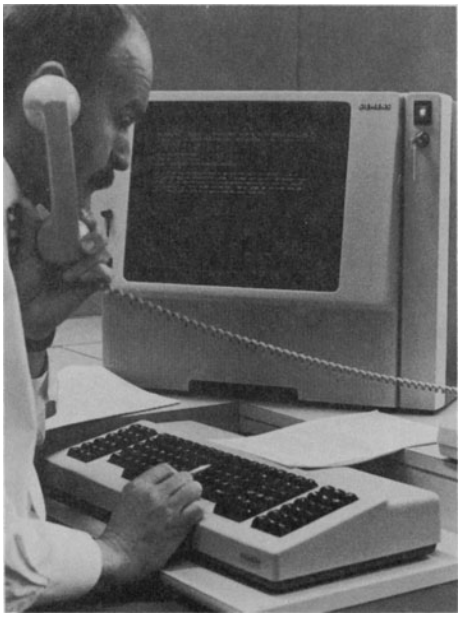
IBM Pressefoto
250572

Tafel 13



Die IBM Serie /1 ist eine Familie kompakter Universalrechner, die sich besonders durch ihren modularen Aufbau und durch umfangreiche Ein- /Ausgabemöglichkeiten auszeichnet. Zum Aufbau von individuellen Konfigurationen stehen unterschiedliche Prozessor-Moduln, Moduln mit Magnetplatten- und Diskettenspeicher, ein Modul zur Ein- /Ausgabe von digitalen und analogen Signalen, sowie Datensichtgeräte, Matrixdrucker und Zeilendrucker zur Verfügung.

IBM Pressefoto
200677/1–6



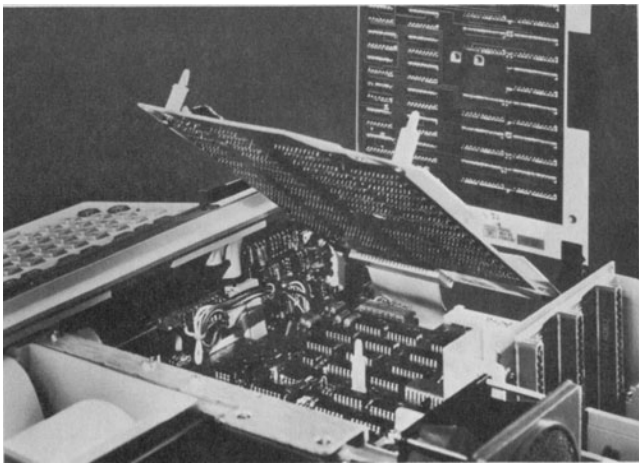
Bei dem neuen EDV-Redaktionssystem COSY 200 von Siemens werden die Texte von Anfang an direkt in den Speicher der Datenverarbeitungsanlage geschrieben. Nach Abruf auf den Bildschirm eines Datensichtgeräts können sie dann papierlos redigiert, mit typografischen Anweisungen versehen und sofort in die Setzmaschine gegeben werden.

siemens pressebild

Tafel 14

Bis zu 64 Benutzer können mit dem neuen 32-Bit-Computersystem VAX-11/780 von Digital Equipment gleichzeitig arbeiten. Als Mehrbenutzer-Multi-Programming-System kann VAX-11/780 in Datenverarbeitungszentren, großen Labors, im Bildungswesen, in Produktionsbetrieben und bei Forschungsgesellschaften eingesetzt werden.

Digital Equipment Pressefoto



Aufklappbare gedruckte Schaltkarten ermöglichen einen leichten Zugang in das Innere des programmierbaren Tischrechners HP 9825A.

Hewlett-Packard Werkfoto

Tafel 15

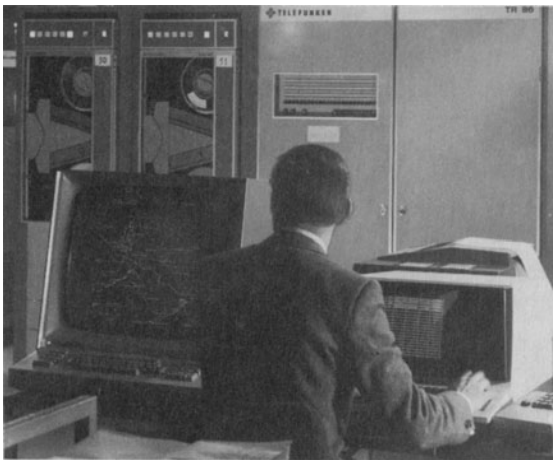
In der GUT-Zentrale in Eschborn/Taunus arbeitet ein Doppelprozessorsystem UNIVAC 1106, an den alle im gesamten Bundesgebiet befindlichen Urlaubsberatungs-Center von GUT-Reisen mittels Bildschirmgeräten angeschlossen sind. Das EDV-System ist in allen Phasen einer Reise, von der Zusammenstellung des Touristikangebots bis zur Rückkehr aus dem Zielgebiet, rund um die Uhr betriebsbereit.

Sperry-Univac-Foto

Platzbuchung bei der Deutschen Lufthansa über Datensichtstationen UNISCOPE 100

Sperry-Univac-Foto

Tafel 16



Datenendgerät AEG/Telefunken

AEG-Telefunken-Werkfoto 1–6702/5L

EDV-System Kienzle 6100

Im Taunus-Zentrum Eschborn bei Frankfurt/M. der Deutschen Bank AG wird die Sorten- und Edelmetallabrechnung (Münzen, Goldbarren etc.) für alle Niederlassungen in der Bundesrepublik auf einem EDV-System Kienzle 6100 durchgeführt. Drei Konsolen und drei Drucker sind online mit der Zentraleinheit des Plattensystems verbunden. Priorität hatten bei diesen mustergültigen Installation, einer typischen Insellösung im Großunternehmen, die Forderungen nach Funktionssicherheit und Sofortverfügbarkeit.

Kienzle-Pressefoto

Tafel 17

COSYNA – das Computersystem für den niedergelassenen Arzt zur Lösung administrativer und organisatorischer Aufgaben in der Praxis, wird auf der neuentwickelten Datenverarbeitungsanlage NCR 8130 eingesetzt.

Foto: NCR 720/3

Neue organisatorische Konzepte eröffnet das Nixdorf-Datentelefon. Diese Anlage bietet nicht nur komfortablen Fernsprechbetrieb, sondern ermöglicht darüber hinaus von jedem Arbeitsplatz aus den direkten Dialog mit dem Computer. Mit dem Datentelefon können sowohl Daten übermittelt als auch Informationen abgefragt werden, wobei der unkontrollierte Zugriff auf zentrale Datenbestände programmgesteuert verhindert werden kann.

Nixdorf Pressebild

Tafel 18

Werkfoto Honeywell/Bull

Terminalcomputer Honeywell/Bull 61/40 mit einem MOS-Speicher, Magnetplatten, einer Diskettenstation, Drucker und Bildschirmgeräten. Das Softwareprogramm umfaßt Standardanwendungen für Handel und Industrie.

Terminalprozessor 7502/15 der ICL Deutschland

Ein Bildschirm/Dialogsystem mit Hardcopy-Drucker, Sichtgerät und rechts im Bild das integrierte Floppy-Disk. Das Modell 7502/15 bietet Anwendern Off-Line-Datenerfassung und Dialogverarbeitung mit dem Hauptcomputer.

Foto: ICL Deutschland

Tafel 19

Carl Schneider

Datenverarbeitungs-Lexikon

2. Auflage

388 Seiten ISBN 3 409 31834 8

Welche Aufgaben kommen dem Programmierer, dem Operator und dem Systemanalytiker zu?

Was ist „tele-processing", „time-sharing" und „real-time"?

Auf diese oder ähnliche Fragen aus dem Bereich der Elektronischen Datenverarbeitung gibt das „Datenverarbeitungs-Lexikon" kurze, aber erschöpfende Antworten. Mit seinen rund 3000 alphabetischen geordneten Begriffen und deren ausführliche Erläuterungen gibt Ihnen das EDV-Lexikon einen raschen Überblick über die Definitionen dieser Fachsprache. Die Erläuterungen werden zusätzlich ergänzt durch:

- ein Verzeichnis mit 380 Abkürzungen für Fachausdrücke und Firmennamen,
- ein englisch-deutsches Glossar mit 700 Begriffen,
- eine Zusammenstellung der wichtigsten Daten der Hardware und Software von 24 Systemfamilien,
- Symboldarstellungen, Ausbildungs- und Weiterbildungsstätten für den Datenverarbeitungskaufmann,
- eine umfangreiche Zeittafel der Datenverarbeitung.

GABLER
Verlag für Wirtschaft

Postfach 15 46
6200 Wiesbaden 1